TAMPEREEN TEKNILLINEN YLIOPISTO

JARI EEROLA

COGNOS SERIES 7 -YMPÄRISTÖN AUTOMATISOINTI COGNOS-MAKROJEN AVULLA

Diplomityö

Tarkastaja: professori Tommi Mikkonen
Tarkastaja ja aihe hyväksytty
Tieto- ja sähkötekniikan tiedekunta-neuvoston kokouksessa 9. toukokuuta 2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Tietotekniikan koulutusohjelma
EEROLA, JARI: Cognos Series 7 -ympäristön automatisointi Cognos-makrojen avulla
Diplomityö, 44 sivua, 12 liitesivua
Toukokuu 2013
Pääaine: Ohjelmistotuotanto
Tarkastaja: professori Tommi Mikkonen
Avainsanat: Cognos Series 7, makro, automatisointi, Access Manager, Impromptu

Ohjelmistoalalla tehdään säännöllisesti toistuvia työtehtäviä erilaisia sovelluksia käyttäen. Työtehtävät saattavat sisältää manuaalisia työvaiheita, jotka mahdollisesti voisi automatisoida. Automatisoinnin avulla yritys pystyy tehostamaan omia työprosessejaan, parantamaan tuottavuutta sekä tarjoamaan asiakkailleen parempaa palvelun laatua. Työtehtävien automatisointi ei kuitenkaan aina ole helppoa. Se edellyttää tuntemusta yrityksen työprosesseista sekä automatisoitavasta ympäristöstä. Lisäksi tarvitsee mahdollisesti perehtyä uusiin teknologioihin, jotka tukevat sovellusten automatisointia.

Tässä diplomityössä käsitellään CGI:lle toteutettua automatisointiratkaisua IBM:n Cognos Series 7 -ympäristöä varten. Diplomityön tavoitteena on selvittää, miten Cognos-makrot soveltuvat automatisoinnin toteuttamiseen sekä esitellä Cognos-makrokieli.

Työn ensimmäisessä teoriaosuudessa käsitellään yleisesti automatisoinnin hyviä ja huonoja puolia sekä esitellään työkaluja liiketoimintaprosessien automatisointiin. Toisessa teoriaosuudessa lukija tutustutetaan Cognos Series 7 -tuoteperheeseen. Tämän jälkeen käydään läpi CGI:n Cognos Series 7 -tuoteperheeseen liittyvät työprosessit sekä esitellään Cognos-makrokieli. Lopuksi pohditaan hyötyjä, joita CGI saa automatisoinnista.

Diplomityö osoittaa, että Cognos-makrot soveltuvat automatisointiin. Makrojen tukena kannattaa kuitenkin hyödyntää muita teknologioita parantamaan käytettävyyttä ja ylläpidettävyyttä. Työ osoittaa myös sen, että automatisointi tuo yritykselle todellisia hyötyjä. CGI:n tapauksessa näitä ovat muun muassa säännöllisten työtehtävien nopeampi valmistuminen sekä työntekijöiden työkuormituksen väheneminen.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Information Technology
EEROLA, JARI: Automatization of Cognos Series 7 environment with Cognos macros
Master of Science Thesis, 44 pages, 12 Appendix pages
May 2013
Major: Software Engineering
Examiner: Professor Tommi Mikkonen
Keywords: Cognos Series 7, macro, automatization, Access Manager, Impromptu

Work at IT industry often includes regular tasks performed with various applications. Tasks may be repetitive and have a lot of manual work phases. There is a chance that some of the work phases could be automated. This may help the company to improve its work processes, increase productivity and offer a better service quality to customers. However, to automate tasks is not always easy. It requires knowledge of company's work processes and the environment where automation is implemented. In addition, automating tasks may require becoming familiar with the technologies that support automating the applications.

This Master of Science Thesis examines the automatization of IBM Cognos Series 7 environment developed for CGI. The objective is to research the suitability of Cognos macros for automation and introduce the Cognos macro language.

This thesis has two theory parts. The first one discusses about the benefits of automation and its possible drawbacks. In addition, it introduces tools for implementing automation. The second theory part introduces Cognos Series 7 product family. After this Cognos macro language and CGI's work processes, related to Cognos Series 7, are discussed about. Finally, the benefit of automation for CGI is pondered.

This thesis shows that Cognos macros are suitable for automating Cognos Series 7 environment. However, using other technologies alongside should be considered. The thesis also proves that a company can get real benefits by automating tasks. For example, in the case of CGI, automation supports completing the regular tasks faster and reduces the workload of employees.

ALKUSANAT

Tätä diplomityötä lähdettiin alun perin työstämään Logicalle vuonna 2012. Ennen kuin diplomityö ehti valmistua, kanadalainen IT-palvelualan yritys CGI osti Logican, ja Logica-nimi vaihtui CGI:ksi. Haluan kiittää molempia yrityksiä mahdollisuudesta tehdä tämä diplomityö heille.

CGI:n puolelta diplomityön tarkastajana toimi järjestelmäasiantuntija Outi Vartiainen. Haluan kiittää häntä perehdyttämisestä minut Cognos-maailmaan sekä saamistani kommenteista diplomityöhön liittyen.

TTY:n puolelta diplomityön tarkastajana toimi professori Tommi Mikkonen. Kiitän häntä diplomityön ohjauksesta sekä saadusta rakentavasta palautteesta. Osittain kiitos kuuluu hänelle myös siitä, että diplomityön kirjallinen osuus valmistui ennen kesää.

Yleisesti haluan kiittää kaikkia työtovereitani, joiden kanssa olen päässyt työstämään erinäisiä projekteja. Niistä on oppinut paljon, ja ne ovat osittain vaikuttaneet tämän diplomityön sisältöön. Lisäksi kiitän vanhempiani, jotka edesauttoivat diplomityöni edistymistä tarjoamalla sopivan kirjoitusympäristön heidän luonaan.

Tampereella 22.4.2013

Jari Eerola

SISÄLLYS

TERMIT JA NIIDEN MÄÄRITELMÄT

Access Manager	Työkalu käyttöoikeuksien ja tietoturvan hallintaan Cognos-työkalujen kanssa.
Automatisointi	Tietokoneen suorittama toimenpide.
BPEL	Lohkorakenteinen xml-pohjainen kieli.
CognosScript Editor	Tekstieditori makrojen kirjoittamiseen ja kääntämiseen.
Cognos Series 7	IBM:n omistama liiketoimintatiedon hallintajärjestelmä.
Dimensio	Bisnestiedon osa-aluetta kuvaava tietojoukko.
ERP	Toiminnanohjausjärjestelmä.
Impromptu	Tietokannan raportointityökalu.
Katalogi	Sisältää metadataa tietokannasta ja näin ollen mahdollistaa tietojen hakemisen Impromptu-raportille.
Kertaluonteinen raportti	Pyynnöstä toimitettava raportti.
Kuutio	Tiedosto, joka mahdollistaa porautumisen.
Makro	Joukko komentoja, joilla suoritetaan sovelluskohtaisia käskyjä.
Petri-verkko	Kaksiosainen graafi, jossa solmut joko ovat paikkoja tai siirtymiä.
PowerPlay Enterprise - Server Administration	Kuution julkaisussa käytetty työkalu.
PowerPlay for Web	Kuutioiden analysointiin tarkoitettu työkalu Upfront-verkkoportaalissa.
PowerPlay for Windows	Kuutioiden analysointiin tarkoitettu työkalu Windows-ympäristössä.
PowerPlay Transformer	OLAP-mallinnustyökalu kuutioiden rakentamiseen.
Report Administration	Vakioraporttien julkaisussa käytettävä työkalu.
Server Administration	Työkalu Upfront-verkkoportaalin konfigurointiin ja monitorointiin.
Upfront	Verkkoportaali, jossa asiakkaat pystyvät tarkastelemaan vakioraportteja ja kuutioita.
Vakioraportti	Upfront-verkkoportaaliin julkaistu raportti.
Valinta	Impromptu-raportin ajohetkellä määritettävä parametri.
YAWL	Avoimeen lähdekoodiin perustuva liiketoimintaprosessien hallintajärjestelmä.

1 JOHDANTO

Ohjelmistoalalla käytetään erilaisia sovelluksia suoritettaessa säännöllisesti toistuvia työtehtäviä, kuten tietokantalatauksia tai raportointia. Työtehtävät saattavat sisältää manuaalisia työvaiheita, jotka mahdollisesti voisi automatisoida. Tämä helpottaa ja nopeuttaa työtehtävien suorittamista. Samalla työntekijöiden kiire vähenee ja heistä aiheutuvien inhimillisten virheiden määrää voidaan pienentää. Liiketoiminnallisesta näkökulmasta tarkasteltuna automatisointi edistää tuottavuutta. Sen seurauksena työntekijöille jää enemmän aikaa tehdä muuta laskutettavaa työtä.

CGI:lle (http://www.cgi.com) on aiemmin toteutettu automatisointiratkaisu Oracle-tietokantalatauksia varten. Automatisointi on todettu hyödylliseksi, koska se muun muassa mahdollistaa tietokantalatausten ajastamisen, niissä käytettävien csv-tiedostojen automaattisen versioinnin sekä parantaa virhetilanteiden havaitsemista. Lisäksi säännöllisten työtehtävien suorittaminen koetaan mieluisemmaksi ja vähemmän työllistäväksi, kun manuaaliset työvaiheet vähenevät.

Onnistuneen automatisoinnin myötä CGI:llä toivottiin helpotuksia IBM:n Cognos Series 7 liittyviin työtehtäviin. Ennen kuin tämä oli mahdollista, tuli löytää keino, miten Cognos Series 7 -sovelluksia voisi automatisoida. Makrot vaikuttivat käyttökelpoiselta vaihtoehdolta. Makrot koostuvat joukosta komentoja, joilla suoritetaan sovelluskohtaisia käskyjä [1, s. 11].

Tässä diplomityössä käsitellään Cognos-makroja, joilla oli keskeinen rooli IBM:n Cognos Series 7 -sovelluksien automatisoinnissa. CGI:lle toteutettavan automatisoinnin tavoitteena oli vähentää manuaalisia työvaiheita, jotta säännöllisesti toistuvat työt kuormittaisivat työntekijöitä vähemmän. Cognos-makrojen lisäksi toteutuksessa hyödynnettiin vba-makroja ja C#-ohjelmointikieltä. Vba-makroja tarvittiin Excel-konversioissa, ja C# toimi käyttöliittymän ohjelmointikielenä. Diplomityössä keskitytään kuitenkin lähinnä Cognos-makroihin. Cognos-makrokielen esittelyn ohella työntavoitteena on tuoda ilmi automatisoinnista saatavia hyötyjä ja samalla tuoda esille, minkälaisia tekijöitä automatisoinnissa kannattaa huomioida. Huonosti suunniteltu ja toteutettu automatisointi ei palvele pitkällä tähtäimellä.

Diplomityön rakenne on seuraava. Luvussa 2 käsitellään automatisoinnista saatavia etuja, mutta myös sen mahdollisesti tuomia haittapuolia. Lisäksi luvussa käsitellään erilaisia työkaluja, joita voidaan käyttää automatisoinnin toteuttamisessa. Cognos Series 7 -sovelluksia käsitellään 3. luvussa. 4. luku on koodipainotteinen, ja siinä esitellään, minkälaisia automatisointeja toteutettiin Cognos-makroilla. 5. luvussa pohditaan tehtyjen ratkaisuiden onnistumista työkalukohtaisesti sekä esitetään, minkälaisia etuja CGI

automatisoinnista sai. Lopuksi 6. luvussa kootaan työtä yhteen ja mietitään työn onnistumista kokonaisuudessaan.

2 AUTOMATISOINTI

Parasuraman & Riley [2, s. 231] määrittelevät automatisoinnin tarkoittavan toimenpidettä, jonka suorittaa joku muu kuin ihminen. Tässä diplomityössä käytetään vastaavanlaista määritelmää. Automatisointi on tietokoneen suorittama toimenpide.

Automatisoinnin avulla voidaan muun muassa helpottaa säännöllisten ja toistuvien työtehtävien suorittamista. Ideaalitapauksessa aikaa vievät, mutta loogiset työtehtävät valmistuvat itsenäisesti ilman käyttäjän syötettä. Näin ei kuitenkaan aina ole, vaan käyttäjää saatetaan tarvita joissain työtehtävään liittyvissä vaiheissa riippuen automatisointiasteesta. Esimerkiksi Parasuraman & Riley [2, s. 231] mukaan on harvinaista, että päätöksien tekoa tai luovaa ajattelutyötä vaativat työtehtävät olisi jätetty automatisoinnin vastuulle.

Sama työtehtävä voidaan toteuttaa eri automatisointiasteella. Työtehtävänä voisi vaikkapa olla csv-tiedostojen lataus tietokantaan usealle asiakkaalle samana päivänä. Jotta sql-lauseita ei tarvitsisi ajaa yksi kerrallaan, työtehtävän helpottamiseksi on rakennettu komentojono. Käyttäjän tarvitsee vain siirtää csv-tiedostot oikeaan kansioon ja käynnistää komentojono tietyllä parametrilla. Yhden tietokantalatauksen loputtua käyttäjä käynnistää seuraavan komentojonon toiselle asiakkaalle. Työtehtävän suorittamista voisi entisestään helpottaa suunnittelemalla käyttöliittymän, joka tukee tiedostojen siirtoa ja komentojonojen käynnistämistä. Vielä automatisoidumpi ratkaisu olisi, että tiedostojen siirto ja komentojonojen käynnistyminen tapahtuisi ajastetusti. Lisäksi komentojonolle välitettävän parametrin arvo pääteltäisiin ajopäivämäärästä.

Kun yrityksessä pohditaan automatisointiratkaisun rakentamista, tulee miettiä, mitä automatisoinnilla on tarkoitus saavuttaa. Mitä mahdollisia etuja ja haittoja automatisoinnista seuraa? Näitä käsitellään kohdissa 2.1 ja 2.2. Lisäksi kohdassa 2.3 käsitellään erilaisia automatisointityökaluja.

2.1 Automatisoinnin hyvät puolet

Automatisointi mahdollistaa rutiininomaisten työtehtävien siirtämisen tietokoneen vastuulle. Kun tietokone toistaa yksinkertaiset ja tylsät työtehtävät, voivat työntekijät keskittyä samanaikaisesti haasteellisempiin ja ehkäpä näin ollen myös mielekkäämpiin työtehtäviin. Lisäksi ihmiseltä kuluu rutiinien toistamiseen suhteellisesti enemmän aikaa kuin tietokoneelta, joten automatisoinnin myötä ehditään tehdä enemmän työtä samassa ajassa. Esimerkiksi yhdessä vaiheessa CGI:ssä oli tarve hakea manuaalisesti csv-tiedostoja toiselta palvelimelta ja uudelleen nimetä ne tietyn logiikan mukaisesti. Tarkoitusta varten tehty tietokoneohjelma suoriutuu työtehtävästä muutamassa sekunnissa, mutta ihmiseltä menee siihen aikaa noin kymmenen minuuttia. Koska csv-tiedostojen

uudelleen nimeämistä täytyy tehdä useita kertoja kuukaudessa, se ei ole kaikista motivoivin työtehtävä.

Ihmiset eivät ole parhaimmillaan suorittaessaan toisteisia työtehtäviä. Heillä on tapana tehdä asioita varioiden, vaikka se ei olisi tarpeen. Tämä saattaa johtaa virheisiin: syötteet ovat virheelliset, työvaiheita saatetaan ohittaa eikä työtehtävään keskitytä täysillä. Automatisointi sen sijaan soveltuu erinomaisesti toisteisiin työtehtäviin. [3, s. 71.] Parasuraman & Riley [2, s. 235] mukaan automatisoinnin eräs keskeinen tavoite onkin vähentää ihmisestä aiheutuvia virhemahdollisuuksia. Breton & Bossé [4, s. 1] sanovat, että virheiden minimointi on erityisen tärkeää monimutkaisissa ympäristöissä, joissa ihmisten tekemillä virheillä saattaa olla kohtalokkaat seuraukset.

Berghammer tutki automatisoinnin hyötyjä pienen kirjaston näkökulmasta. Taloudellisesta näkökulmasta hän totesi, että automatisoinnin rakentaminen ja ylläpitäminen vaatii kulunsa. Ilman automatisointia henkilöstöä pitäisi kuitenkin tulevaisuudessa palkata lisää. Automatisointi toisi säästöjä henkilöstö- ja palkkakuluihin, koska muun muassa aikaa vievien työtehtävien kestoa voitaisiin pienentää. [5, s. 62-67.] Lisäksi myöhästymismaksujen keräämisessä tapahtuisi 50% kasvu. Automatisointi toisi kirjastolle myös muita kuin taloudellisia hyötyjä. Se mahdollistaisi tiedon jakamisen toisten kirjastojen kanssa ja tukisi näin ollen yhteistyöprojekteja. [5, katso 6, s. 81.]

Automatisoinnin myötä työtehtävistä saattaa tulla helpommin lähestyttäviä. Mikäli tiimiin liittyy uusi työntekijä, hänen ei välttämättä tarvitse heti tietää, mitä kaikkea on työprosessin taustalla. Automatisointi voi tukea työntekijää opastamalla häntä suorittamaan työtehtävään liittyvät työvaiheet. Esimerkiksi muuan CGI:ssä vastaan tullut työprosessi on sen verran monivaiheinen, mistä johtuen uuden työntekijän voi olla vaikea muistaa, mitä pitää tehdä missäkin järjestyksessä. Tämän vuoksi tueksi on rakennettu käyttöliittymä, joka näkyy kuvassa 2.1. Käyttöliittymän avulla voidaan käynnistää komentojonoja, jotka suorittavat taustalla olevat työvaiheet. Tässä tapauksessa automatisointi mahdollistaa asiantuntijatehtävien suorittamisen keneltä tahansa. Lisäksi automatisoinnin myötä lomien aikatauluttaminen on helpompaa, sillä asiantuntijalle löytyy helpommin sijainen suorittamaan säännölliset työtehtävät.

Automatisointi on keino parantaa asiakkaalle tarjottavan palvelun laatua. Esimerkiksi CGI:lle on rakennettu automatisointiratkaisu, joka huolehtii tietokantalatauksista. Asetustiedoston avulla on mahdollista määrittää asiakaskohtaisesti, mitkä csv-tiedostot tulee löytyä ennen kuin tietokantalataus voidaan aloittaa. Mikäli jokin tiedosto puuttuu, nämä listataan lokiin ja tietokantalataus estetään. Loki parantaa latausprosessin seurattavuutta. Esimerkiksi lokista selviää, milloin tietokantalataus on alkanut ja millä parametreilla se on käynnistetty. Loki tukee myös virhetilanteiden selvittämistä, sillä siitä voi tarkistaa, onko lataus jäänyt jumiin tai miten tietokantataulujen rivimäärät ovat muuttuneet latauksen jälkeen. Jos esimerkiksi kahden peräkkäisen kuukauden rivimäärät ovat likimain samat, tämä saattaa kertoa siitä, että tietokannalle varattu tila on päässyt loppumaan. Tutkimalla automaattisesti luotavaa lokia voidaan epäonnistunut lataus havaita ennen kuin asiakas ilmoittaa siitä.

Kuva 2.1. *Käyttöliittymä automatisoinnin tukena.*

CGI:ssä on todettu, että tietokantalatausten automatisoinnin myötä uusintalataukset on nopeampi tehdä, koska tietokantalatauksia voidaan ketjuttaa. Kun ensimmäinen lataus päättyy, seuraava alkaa. Eduksi on havaittu myös se, että tietokantalataukset voi ajastaa suoritettavaksi ilta-aikaan tai viikonloppuna. Asiakkaalla käytössä olevaa raportointijärjestelmää ei pysty käyttämään tietokantalatauksen aikana, joten se on paras tehdä muulloin kuin päiväsaikaan. Työntekijät hyötyvät myös ajastusmahdollisuudesta, sillä jos tietokantalataukset on sovittu tehtävän viikonloppuna, heidän ei automatisoinnin myötä tarvitse käydä lauantaina käynnistämässä tietokantalatausta.

2.2 Automatisoinnin huonot puolet

Automatisointiratkaisun rakentaminen ei tapahdu hetkessä. Ensinnäkin se vaatii ymmärrystä, miten työtehtäviä tehdään. Työprosessin opetteluun tarvitsee varata aikaa. Toiseksi pitää tuntea ympäristö ja mahdollisesti tutustua uuteen teknologiaan. Esimerkiksi tietyn järjestelmän automatisointi saattaa edellyttää kyseisen järjestelmän makrokielen opettelua. Tämä vaatii teknologisesti soveltuvan henkilön. Jos tällaista ei tiimistä valmiiksi löydy, saattaa automatisointi jäädä ajatustasolle. Mikäli teknologia on vieras, pitää sen opetteluun varata riittävästi aikaa. Hyötyjä ei saada välittömästi, vaan ratkaisun kehittely ja mahdollisesti iterointi vaatii kulunsa. Ympäristö saattaa myös asettaa joitain rajoitteita. Esimerkiksi, jos automatisointi aiotaan toteuttaa .Net Framework –teknologialla, pitäisi varmistua, että lopullinen ympäristö tukee .Net Framework –

ohjelmien suorittamista. Ei välttämättä ole itsestään selvää, että ympäristön mukaiselle palvelimelle voidaan asentaa tai päivittää .Net Framework, mikäli palvelimen ylläpitovastuu on toisella yrityksellä.

Automatisoinnin voi tehdä huonosti. Jos esimerkiksi automatisointi edellyttää tiedostojen käsittelyä, pitäisi joustavasti pystyä ilmoittamaan tiedostojen sijainnit. Ylläpidettävyys on heikkoa, mikäli polut on kovakoodattu käännettyyn lähdekoodiin. Vaikka lähdekooditiedostot olisivat vapaasti muokattavissa, saattaa tämäkin olla haaste. Niitä ei välttämättä ole kommentoitu riittävällä tasolla. Dokumentaatio saattaa myös olla puutteellinen. Tämän johdosta samankaltaisen automatisointiratkaisun käyttöönottaminen toisessa ympäristössä voi osoittautua vaikeaksi. Dokumentoinnin pitäisikin kuulua oleellisena osana automatisointia, sillä sen alkuperäinen kehittäjä ei välttämättä ole jatkossa ylläpitämässä automatisointiratkaisuaan. Huomionarvoista on myös se, että automatisoinnin ylläpidosta aiheutuu kustannuksia yritykselle.

Dokumentoinnin ohella versiointi pitäisi muistaa. Jos samalta palvelimelta löytyy useita kopioita automatisointiratkaisusta, uuden työntekijän voi olla vaikea tietää, mitä niistä pitäisi käyttää. Myöhempiin versioihin on saatettu tehdä tärkeitä korjauksia. Riskinä on myös se, että vanhaan versioon lähdetään tekemään muutoksia. Jos tämä huomataan jälkikäteen, kuluu ylimääräistä työaikaa, kun muutokset integroidaan uusimpaan versioon.

Automatisointiratkaisu voi olla joustamaton. Esimerkiksi tietokantalatauksessa käytettävät parametrit voivat perustua komentojonon ajohetkeen. Jos uusintalataus tarvitsee tehdä jälkikäteen, joutuuko tietokantalatauksen tekemään manuaalisesti. Aikaan sidottu parametrien päättely ei tästä johtuen välttämättä ole hyvä ratkaisu. Toisaalta kaikkia parametreja ei välttämättä voi edes päätellä luotettavasti, joten automatisoinnilla on rajoitteensa. Yksi este manuaaliselle uusintalataukselle voi myös olla, ettei kukaan osaa tehdä sitä. Bainbridgen [7, s. 775] mukaan syvällinen osaaminen saattaa kadota automatisoinnin seurauksena. Vanha työntekijä ei välttämättä enää muista, miten lataukset on ennen tehty manuaalisesti. Uusi työntekijä on puolestaan aina käyttänyt automatisointiratkaisua, eikä tiedä muusta.

Automatisointi voi sisältää virheitä. Testaaminen saattaa olla puutteellista, eikä kaikkia erikoistapauksia ole huomioitu. Työntekijät saattavat myös liikaa luottaa siihen, että automatisointi toimii halutulla tavalla. Saattaa myös olla, että he käyttävät automatisointia väärin, koska käyttö ei ole intuitiivista tai järjestelmän käyttöä ei ole koulutettu tai dokumentoitu riittävällä tasolla. Endsleyn [8] mukaan koulutuksen riittämättömyys ja epäintuitiivinen käyttöliittymä ovatkin merkittäviä kompastuskiviä onnistuneelle automatisoinnille. Virheet automatisoinnissa tai sen käytössä puolestaan saattavat johtaa asiakasreklamointeihin. Asiakkaat eivät ole tyytyväisiä, jos heille toimitetut palkkaraportit ovat virheellisiä. Parasuraman & Riley [2, s. 238] mukaan automatisoinnin käyttöä saatetaan silti jatkaa, jos virhetilanteet ovat harvinaisia.

Endsley väittää automatisoinnin parantaneen useassa tapauksessa tehokkuutta yli ihmiskyvyn, ja seurauksena ihmisen rooli on muuttunut. Sen sijaan, että ihminen tekisi työtehtävät itse, hänen tuleekin seurata niiden valmistumista. Endsley väittää, että rooli

ei ole ihmiselle luonnollinen, sillä ihmiset ovat hitaita havaitsemaan automatisoidun järjestelmän virhetilanteet. Lisäksi kun virhe havaitaan, ihmisellä kuluu aikaa selvittää järjestelmän tila ja perimmäinen syy virheeseen. [8.] Automatisoinnin hyödyntämisessä tulisikin ymmärtää, että automatisoinnin toimivuutta tulisi säännöllisesti seurata. Seurattavuus voi kuitenkin olla haaste, mikäli kunnollista automaattista lokia ei ole saatavana. Vaikka kunnollinen lokisysteemi olisi rakennettu, saattaa olla, että työntekijät eivät seuraa sitä riittävän aktiivisesti, mistä johtuen virheiden havaitseminen vie aikaa. Toinen mahdollisuus on se, että lokia muodostuu liikaa ja siitä on vaikea löytää oleellinen tieto.

Breton & Bossé [4, s. 1-2] mukaan automatisointi voidaan nähdä potentiaalisena ratkaisuna vähentää ihmisten työkuormaa, stressiä ja väsymystä. Seuraukset eivät kuitenkaan aina ole positiivisia. Endsley tutki, miten ihmiset reagoivat teknologiseen muutokseen. Havaintona oli, että automatisoinnin myötä ihmisten elämänlaatu sekä sitoutuminen työhön vähenivät. Osittain tämä johtui muutosvastarinnasta ja kielteisistä asenteista kohti automatisointia. Kolmasosa 32 tutkimukseen osallistuneesta henkilöstä ilmoitti, että heillä on vähäinen kiinnostus mukautua muutokseen. [9, s. 601.] Automatisoinnin käyttöönotossa voi olla siis haasteita. Riskinä saattaa olla, että työntekijät jatkavat mieluummin vanhojen rutiinien toistamista. Tällöin automatisointiin käytetty aika on mennyt hukkaan.

Vaikka automatisointi mahdollistaa työtehtävien suorittamisen nopeammin ja täten laskutettavaa työtä ehtii periaatteessa tehdä enemmän, näin ei välttämättä ole. Jos esimerkiksi ylläpitosopimuksen mukaan laskutus perustuu käytettyyn työaikaan, väheneekö laskutettavat tunnit automatisoinnin myötä? Automatisointia pohtiessa, tuleekin miettiä sen kannattavuutta: tehdäänkö automatisointi ilmaiseksi asiakkaalle vai voiko laskuttaa samat tunnit, vaikka työtehtävien tekemiseen kuluisi vähemmän aikaa?

2.3 Automatisoinnin työkalut

Liiketoimintaprosessien automatisointia tukemaan on kehitetty useita erilaisia työkaluja. Näitä ovat muun muassa erilaiset liiketoimintaprosessien mallinnuskielet, kuten BPEL (*Business Process Execution Language*) ja YAWL (*Yet Another Workflow Language*), sekä makrot.

BPEL on 2003 esitetty lohkorakenteinen xml-pohjainen kieli. Se perustuu web-palveluihin, ja mahdollistaa useiden järjestelmien linkittämisen riippumatta siitä, missä järjestelmät sijaitsevat. BPEL:ssä liiketoimintaprosessit kuvataan itsenäisinä lohkoina, jotka yhdessä muodostavat monimutkaisempia lohkoja. Koska BPEL:ssä on kuitenkin mahdollista määrittää lohkorajoja ylittäviä riippuvuuksia kontrollilinkkien (*control links*) avulla, ei BPEL ole täysin lohkorakenteinen. Kielen ilmaisurajoitteena on se, ettei se tue ihmisten osallistumista liiketoimintaprosesseihin. [10, s. 7; 11.] Tosin BPEL:ään perustuvat integraatiokehitysympäristöt, kuten IBM:n WebSphere Integration Developer, voivat laajentaa BPEL:ää tukemaan myös ihmisvuorovaikutteisia prosesseja [12]. IBM:n ohella BPEL:ää tukevia ohjelmistoja kehittävät muun muassa Microsoft, SAP ja Oracle [11].

Käyttötapaus BPEL:än hyödyntämisestä voisi olla seuraava: Yritys rakentaa ja asentaa sooda-automaatteja ympäri maailmaa. Sillä on käytössä ERP (*Enterprise Resource Planning*) -toiminnanohjausjärjestelmä, jossa kaikki ylläpito tehdään. Kun uusi sooda-automaatti valmistetaan, se rekisteröidään ERP-järjestelmään. Työntekijän täytyy tällöin asentaa automaattiin maakohtainen järjestelmäversio. Tätä varten yritys on ostanut palvelimen, jota voidaan käyttää järjestelmäversion asentamiseen. ERP-järjestelmän pitäisi kuitenkin ensin pystyä keskustelemaan palvelimen kanssa. ERP-järjestelmä ja palvelin voisivat keskustella toistensa kanssa suoraan. Jos palvelin päässä kuitenkin tarvitsee tehdä asetusmuutoksia, ne pitää todennäköisesti huomioida myös ERP-järjestelmässä. Tämä saattaa vaatia ERP-järjestelmän uudelleen kääntämistä ja asentamista, mikä ei välttämättä ole toivottavaa. Ongelma ratkaistaan käyttämällä BPEL-palvelinta välissä. ERP kutsuu sooda-automaatteihin yhteydessä olevaa palvelinta BPEL-palvelimen kautta. Mikäli palvelin päässä muuttuu jokin, muutokset huomioidaan BPEL-palvelimella. BPEL-palvelinta voisi hyödyntää myös tilanteessa, jossa myytävien tuotteiden hinnat pitää päivittää. Ennen työntekijän on täytynyt käydä yksitellen päivittämässä hinnat manuaalisesti. Koska sooda-automaatit ovat nyt yhteydessä verkkoon, riittää että hinnat päivitetään ensin ERP-järjestelmään. Sitten ERP-järjestelmä lähettää uudet hinnat BPEL-palvelimen prosessille, joka pyytää sooda-automaatteihin yhteydessä olevaa palvelinta päivittämään hinnat. [11.]

YAWL on ilmainen, avoimeen lähdekoodiin perustuva liiketoimintaprosessien hallintajärjestelmä. Se on lähtökohtaisesti kehitetty Petri-verkkojen pohjalta. Petri-verkko on kaksiosainen graafi, jossa solmut ovat joko paikkoja tai siirtymiä. Paikat kuvataan ympyröillä ja siirtymät neliöillä. Kaaret toimivat yhdistävinä tekijöinä. YAWL:in yhtenä tavoitteena oli laajentaa Petri-verkkojen kuvauskieltä siten, että muun muassa saman tehtävän useiden samanaikaisten instanssien esittäminen olisi helpompaa. YAWL:in mukana tulee graafinen editori, jolla rakennetaan liiketoimintaprosesseista visuaalisia kaavioita. Samalla liiketoimintaprosessit tulevat dokumentoiduiksi. Visuaalisuus tukee myös työnkulun virhetilanteiden havaitsemista. [10, s. 10-11; 13.] Koska YAWL ei ole kaupallinen työkalu, sen huonona puolena voidaan nähdä se, että yritys ei välttämättä saa riittävästi tukea työkalun käyttöön. Bradford & Dumas mukaan liiketoimintaprosessien hallintajärjestelmän käyttöönottaminen saattaakin olla huono ratkaisu, mikäli se ei ole linjassa yrityksen liiketoiminnan kanssa. Tällöin sen käyttö saattaa osoittautua jopa tehottomammaksi kuin manuaaliset prosessit. [13.]

Liiketoimintaprosessien hallintajärjestelmien hyödyntäminen automatisoinnissa edellyttää hallintajärjestelmiin perehtymistä. Eri spesifikaatioiden tutkiminen ja hallintajärjestelmien soveltuvuuden pohtiminen voi viedä aikaa. Makrot saattavat olla hieman kevyempi lähestymistapa automatisointiin olettaen, että käytettävä sovellus tukee niitä. Helpoimmillaan käyttäjä voi hyödyntää makroja ilman ohjelmointitaitoja. Esimerkiksi Excel tukee makrojen nauhoittamista ja toistamista. Tätä ominaisuutta voisi hyödyntää toistuvissa Excel-konversioissa. Toisaalta ohjelmointikokemus on hyödyksi, sillä pelkän nauhoituksen avulla ei välttämättä saa toteutettua haluttua toiminnallisuutta. Kuitenkin nauhoituksen avulla voi generoida osan koodista ja muokata sitä tarpeiden mukaisesti.

Kaikki makrokieliä tukevat ohjelmat eivät kuitenkaan tue nauhoitusta, joten ohjelmointikokemus on hyödyksi.

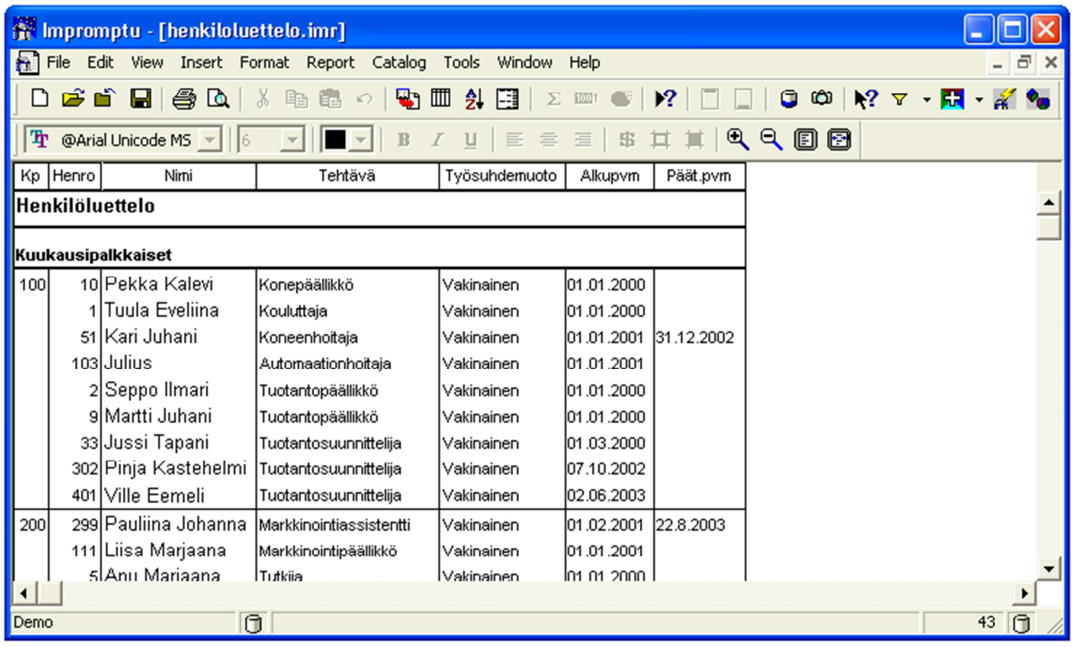
3 COGNOS SERIES 7

Cognos Series 7 on IBM:n omistama liiketoimintatiedon (*business intelligence*) hallintajärjestelmä. IBM hankki oikeudet järjestelmään ostamalla kanadalaisen Cognos-yrityksen vuonna 2007 [14]. Cognos Series 7 -tuoteperhe koostuu muun muassa Impromptu-, Access Manager- ja Upfront -ohjelmistoista.

Cognos Series 7:n tarjoamien työkalujen avulla asiakkaille voidaan tuottaa erilaisia kertaluonteisia raportteja, vakioraportteja tai kuutioita vastaamaan asiakasyritysten raportointitarpeita. Kertaluonteisella raportilla (*ad hoc*) tarkoitetaan pyynnöstä toimitettavaa raporttia. Esimerkiksi on sovittu, että joka kuukauden ensimmäinen päivä toimitetaan tietyt Excel-muotoiset palkkaraportit. Vakioraportilla viitataan raporttiin, jota käyttäjä pääsee tarkastelemaan Upfront-verkkoportaalissa. Kuutio (*PowerCube*) on taas sellainen tiedosto, joka mahdollistaa porautumisen. Tällöin esimerkiksi tarkasteltaessa tietyn yrityksen tulosta voitaisiin aluksi tarkastella vuositulosta. Tämän jälkeen tarkastelu voitaisiin rajata tiettyyn vuosineljännekseen ja lopulta tiettyyn kuukauteen.

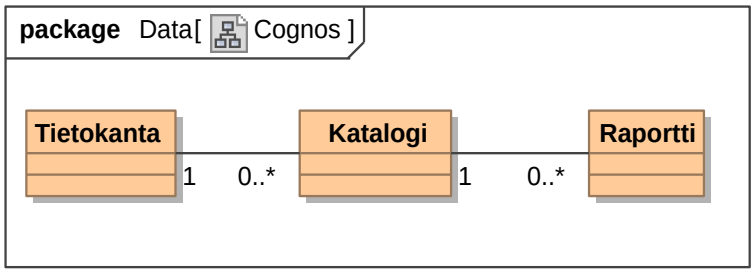
3.1 Impromptu

Impromptu on tietokannan raportointityökalu Windows-ympäristölle [15, s. 3]. Impromptulla pystyy luomaan raportteja sekä hallinnoimaan tietoturvallisuutta esimerkiksi rajoittamalla käyttäjäluokkien pääsy tiettyihin tietokannan tauluihin. Kuvasta 3.1. nähdään, miltä Impromptulla tehty raportti näyttää.



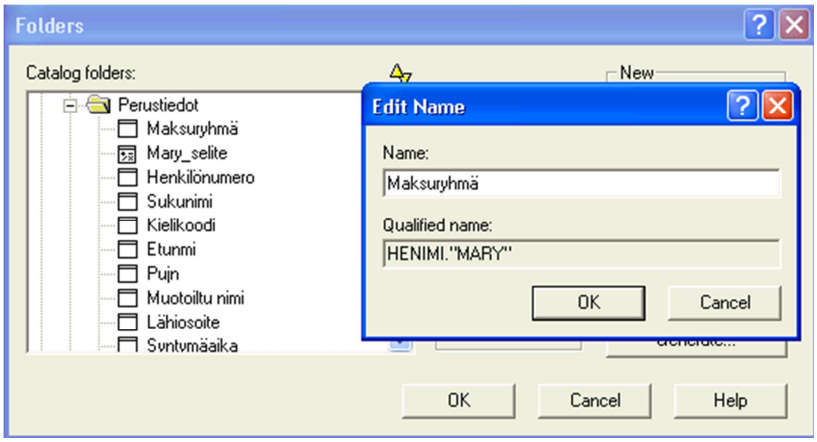
Kuva 3.1. *Impromptulla luotu henkilöluettelo-raportti.*

Impromptu-raportointi edellyttää tietokantayhteyden määrittämistä. Impromptu tukee sekä paikallisia tietokantoja (*local database*) että etätietokantoja (*remote database*). Etätietokantoja pystytään käyttämään joko tietokannan toimittajan ohjelmarajapinnan kautta, kuten Microsoft SQL Serverin Ole DB, tai ODBC-ajurin avulla. [16, s. 139.] Impromptu muodostaa yhteyden tietokantaan katalogin avulla. Katalogi sisältää metadataa tietokannan rakenteesta eli tietokannan nimen ja sijainnin, taulujen väliset liitokset sekä katalogiin valittujen taulujen nimet [16, s. 31]. Raporttiin voi liittyä vain yksi katalogi, mutta eri raportit saattavat hyödyntää yhteistä katalogia [15, s. 4]. Kuva 3.2. havainnollistaa tietokannan, katalogin ja Impromptulla tehtyjen raporttien välistä suhdetta.



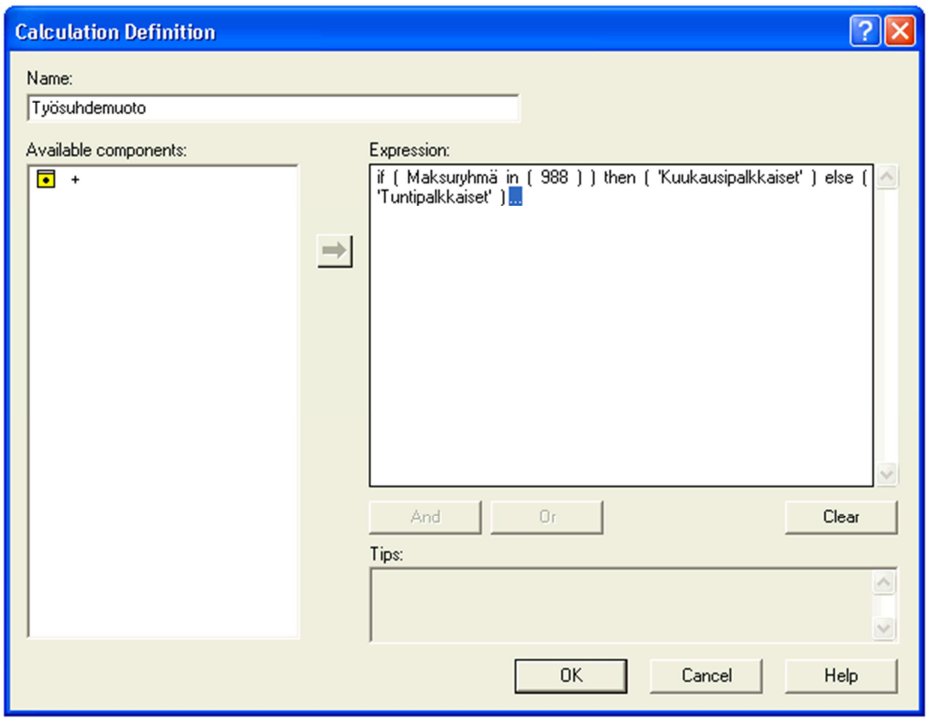
Kuva 3.2. *Katalogin merkitys raportoinnissa.*

Metadatan ohella katalogi koostuu kansioista ja sarakkeista. Sarakkeiden sisältämät arvot saadaan tietokannasta. Esimerkiksi kuvan 3.3. ”Maksuryhmä”-tieto löytyy tietokannan HENIMI-taulusta ja MARY-kentästä.



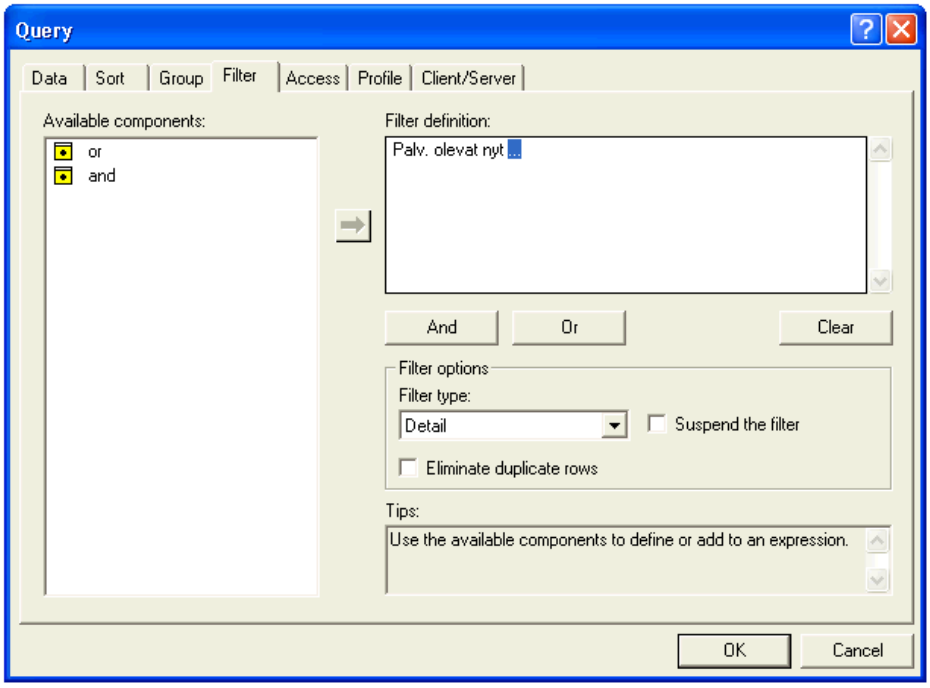
Kuva 3.3. *Katalogin kansiorakenne.*

Lisäksi katalogit voivat sisältää esimääritettyjä laskutoimituksia (*calculation*), ehtoja (*condition*) ja valintoja (*prompt*), joita voidaan raportille lisätä. Yksinkertainen laskutoimitus voisi olla sellainen, että jos maksuryhmä on tietty, tiedetään henkilön olevan kuukausipalkkainen. Muussa tapauksessa kyseessä on tuntipalkkainen henkilö. Päättelyn perusteella tulostetaan raportille joko teksti ”Kuukausipalkkaiset” tai ”Tuntipalkkaiset”, kuten kuvassa 3.1. Kuvasta 3.4. nähdään laskutoimitus.



Kuva 3.4. *Laskutoimituksen määrittäminen.*

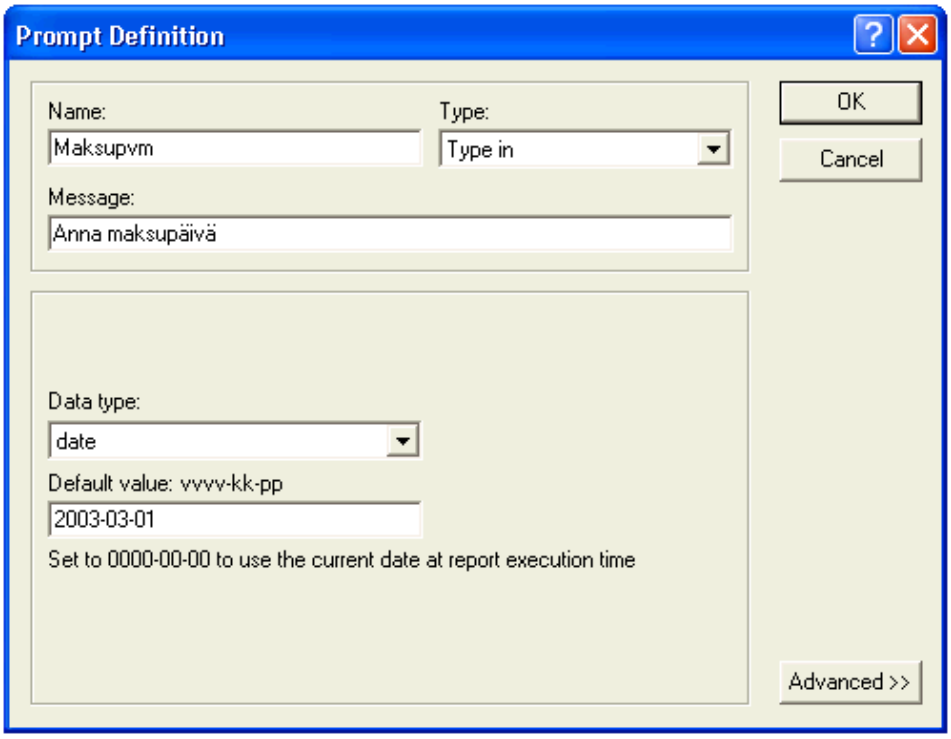
Ehtojen avulla rajataan tulosjoukkoa. Esimerkiksi ehdossa voitaisiin rajata raportin tulosjoukkoa siten, että näytetään vain henkilöt, joilla on työsuhde voimassa. Jos katalogiin määritettyä ehtoa haluaisi hyödyntää, se lisätään osaksi raportin suodatinta (*filter*). Kuvasta 3.5. nähdään, miltä näyttää suodatin, jolle on lisätty ehto.



Kuva 3.5. *Ehdon käyttäminen suodattimessa.*

Valintojen avulla voidaan myös rajata tulosjoukkoa raportin ajohetkellä. Kun raportti ajetaan, saatetaan käyttäjää esimerkiksi pyytää kehotteessa syöttämään kuukauden-

mukainen maksupäivä. Kuvasta 3.6. nähdään, miten tällainen valinta määritetään. Valinnalle syötettyä arvoa voidaan hyödyntää osana laskutoimitusta tai suodatinta. Valinnan ei ole pakko löytyä ennestään katalogista, vaan raporttikohtaisia valintoja pystytään luomaan.



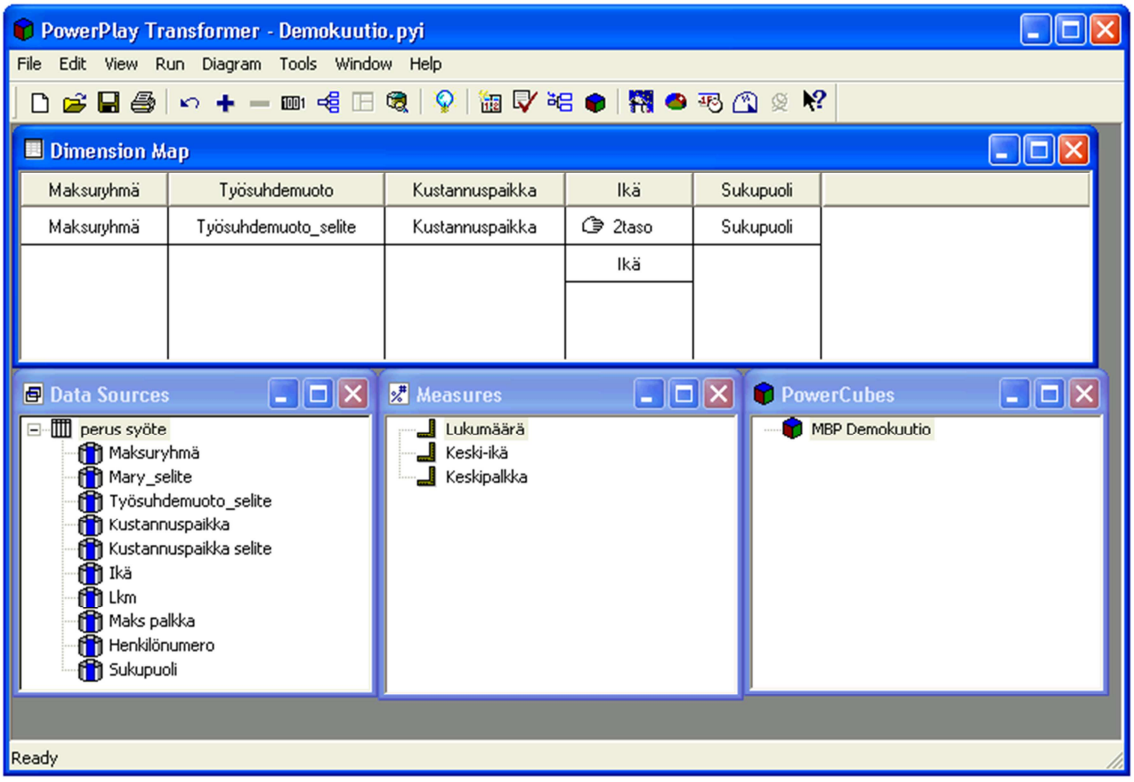
Kuva 3.6. *Valinnan määrittäminen.*

Impromptulla muokataan sekä katalogeja että raportteja. Sen käyttöliittymän avulla imr-päätteiset Impromptu-raportit voidaan myös muuntaa julkaisukelpoisimpiin tiedostomuotoihin, kuten xls, pdf ja html.

3.2 PowerPlay Transformer ja PowerPlay

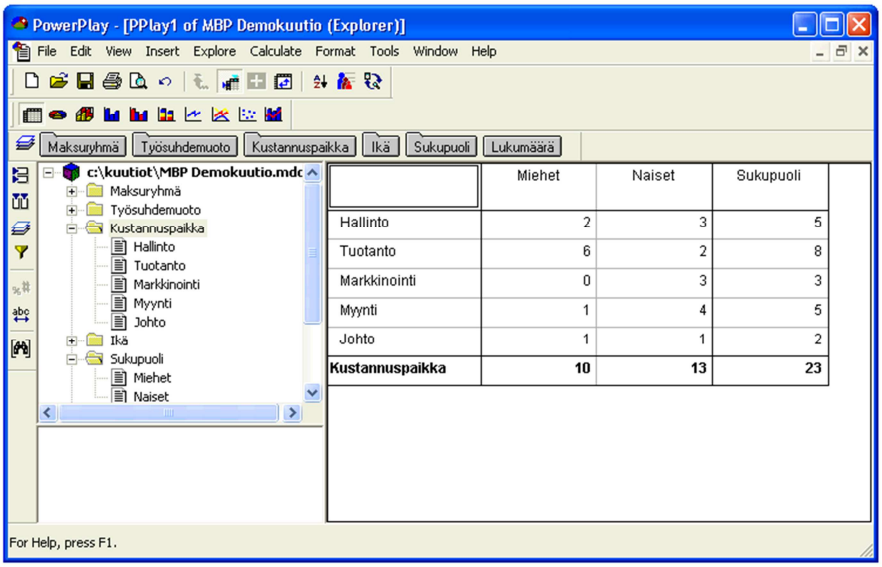
PowerPlay Transformer on OLAP (*Online Analytical Processing*) mallinnustyökalu, jota käytetään rakentamaan monidimensionaalisia malleja, eli kuutioita, tietolähteestä. Termi dimensio (*dimension*) viittaa tietojoukkoon, joka kuvaa bisnestiedon osa-aluetta. Dimensiot vastaavat kysymyksiin kuka, mitä, missä ja milloin. Esimerkiksi kuvassa 3.7. dimensioita ovat maksuryhmä, työsuhdemuoto, kustannuspaikka, ikä sekä sukupuoli. Dimensiot koostuvat hierarkisista kategorioista, jotka ryhmitellään tasoittain (*level*). Samaan dimensioon voi liittyä yhdestä useampaa tasoa. Tasot ja kategoriat mahdollistavat porautumisen. Esimerkiksi tasona voisi olla kustannuspaikka ja siihen liittyvinä kategorioina tuotanto tai markkinointi. Kuution tietolähteenä voivat toimia muun muassa Impromptun iqd-päätteiset kyselymääritystiedostot (*Impromptu query definition*) sekä csv-tiedostot. Kuvassa 3.7. tietolähde "perus syöte" perustuu iqd-tiedostoon. Iqd-tiedostot ovat tekstitiedostoja, jotka sisältävät sql-kyselyitä. Valittavia mittareita (*measures*) ovat lukumäärä, keski-arvo sekä keskipalkka. Mittarit sisältävät numeerista

dataa, jotka pohjautuvat tietolähteeseen tai PowerPlay Transformerilla tehtyihin laskentakaavoihin. Mittarit auttavat muun muassa vastaamaan kysymyksiin kuinka moni tai kuinka paljon. [17, s. 7-10; 18.]



Kuva 3.7. *PowerPlay Transformer ja kuution mallinnus.*

Kun PowerPlay Transformerilla luo kuution pyi-päätteisestä mallitiedostosta tulee mdc-päätteinen kuutiotiedosto. Työpöydällä kuutiotiedostoja pystyy analysoimaan PowerPlay for Windows -työkalulla. Kuvasta 3.8. nähdään, miltä edellä olevasta mallista luotu kuutio näyttää kyseisellä työkalulla tarkasteltuna. Upfront-verkkoportaalista puolestaan löytyy PowerPlay Web -työkalu kuutioiden analysoimiseen.

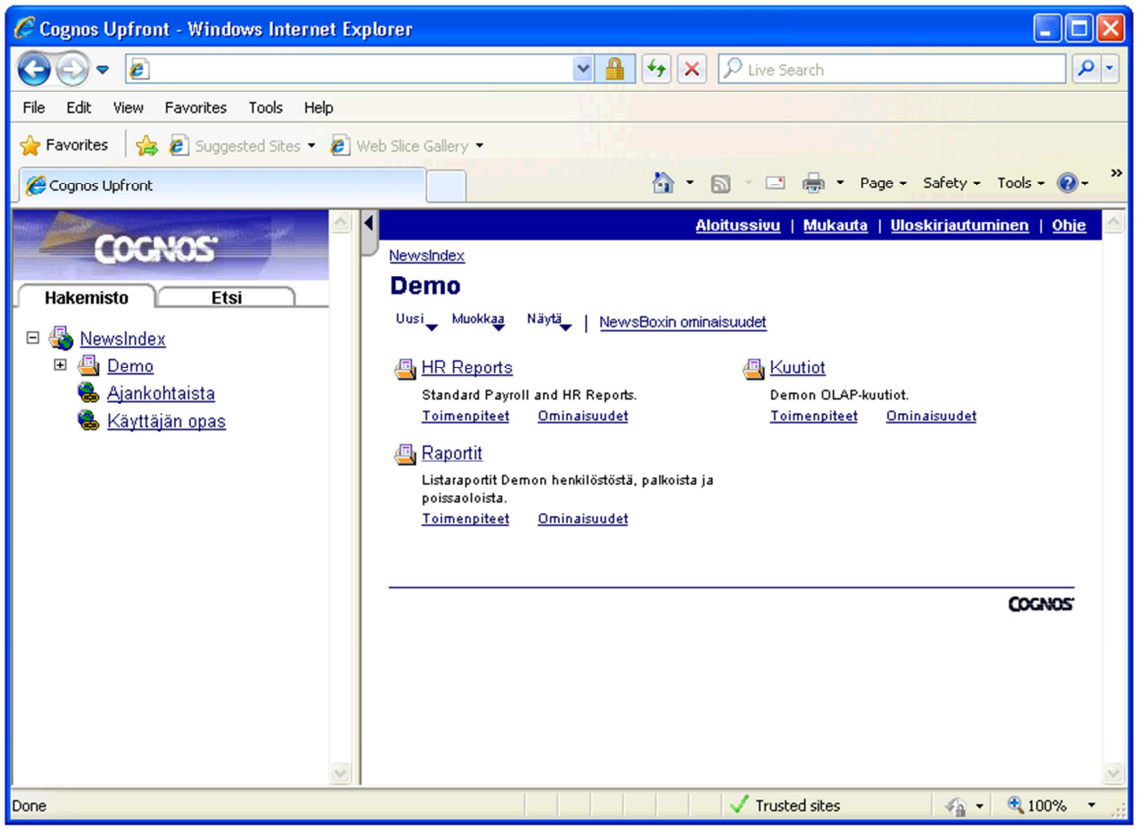


Kuva 3.8. *Kuution analysointi PowerPlay for Windows -työkalulla.*

3.3 Upfront

CGI:llä on toimintaperiaatteena, että Cognos-raportoinnin käyttöönottovaiheessa määritellään asiakkaalle tarjottavat vakioraportit sekä kuutiot. Tyypillisten raporttien joukkoon lukeutuvat henkilöstö- ja palkkaraportit. Raportit julkaistaan Upfront-verkkoportaaliin, josta asiakas pääsee niitä tarkastelemaan.

CGI vastaa Upfront-verkkoportaalin päivittämisestä sovitun aikataulunmukaisesti. Tämä edellyttää tietokantalatauksien ohella katalogin päivittämistä ja julkaisua vakioraporttien tapauksessa sekä kuutioiden päivittämistä että julkaisua kuutioiden tapauksessa. Kuvasta 3.9. nähdään, miltä Upfront-verkkoportaalin käyttöliittymä näyttää.



Kuva 3.9. *Upfront-verkkoportaali.*

3.4 Server Administration

Server Administration on työkalu, jota käytetään konfiguroimaan ja monitoroimaan Upfront-verkkoportaalia [19, s. 7]. Server Administration jakautuu useaan eri työkaluun, mutta säännöllisten raportointityötehtävien kannalta keskeisiä ovat PowerPlay Enterprise - Server Administration sekä Report Administration.

PowerPlay Enterprise - Server Administrationia käytetään kuution julkaisun yhteydessä. Ennen kuin päivitetty kuutio julkaistaan Upfront-verkkoportaaliin, se pysäytetään. Pysäytyksen jälkeen päivitetty kuutio kopioidaan vanhan päälle. Tämän jälkeen kuutio käynnistetään uudestaan. Pysäytys ja käynnistys voidaan tehdä myös komentoriviltä ppadmtool-ohjelmalla.

Report Administrationia käytetään vakioraporttien julkaisuun. Tässä tapauksessa päivitetty katalogi kopioidaan vanhan päälle. Sen jälkeen julkaistaan raporttikokonaisuus Report Administration -työkalulla.

3.5 Access Manager

Access Manager on työkalu määritellä, tallentaa ja ylläpitää tietoturvaa keskitetysti eri Cognos-työkalujen kanssa [20, s. 7]. Access Managerin avulla pystyy muun muassa luomaan uusia käyttäjiä Upfront-verkkoportaaliin sekä määrittämään, millaiset käyttöoikeudet kyseiselle käyttäjätunnukselle annetaan. Esimerkiksi käyttäjälle saatetaan haluta määritellä sellaiset oikeudet, että hänellä on vain oikeus tarkastella vakioraportteja eikä kuutioita ollenkaan. Tämä voi olla haluttua, mikäli lisenssihinta nousee kuutioiden katseluoikeuden myötä. Lisäksi saattaa olla tarve rajoittaa, mitä maksuryhmiä käyttäjä pääsee tarkastelemaan, jotta henkilö ei näkisi ulkopuolisten henkilötietoja.

Käyttäjälle myönnettävät käyttöoikeudet perustuvat käyttäjäluokkiin, joita Access Managerilla luodaan. Tyypillisesti käyttäjäluokka edustaa joukkoa käyttäjiä, joilla on samankaltaisia toiminnallisuuksia organisaatiossa. Käyttäjälle voi olla myös määritetty useita käyttäjäluokkia. Access Managerin avulla käyttäjäluokkakohtaisesti voidaan rajata minä viikonpäivinä käyttäjillä on oikeus päästä tietoon käsiksi. Työkalulla ei kuitenkaan voi määrittää, onko käyttäjällä esimerkiksi oikeus tarkastella tietyn kuution kaikkia dimensioita. Tämä rajaus tulee tehdä PowerPlay Transformerilla kuution luonnin yhteydessä. [20, s. 53-56.]

3.6 Cognos-makrokieli ja CognosScript Editor

IBM:n Cognos-makrokieli (*IBM CognosScript language*) on modulaarinen ohjelmointikieli: koodi on jaettu aliohjelmiin ja funktioihin. Näissä määritetyt muuttujat ovat käytettävissä ainoastaan kyseisessä aliohjelmassa tai funktiossa. Näkyvyysalue voi myös olla moduulikohtainen, jos muuttujan määrittää aliohjelman tai funktion ulkopuolella. Tällöin muuttuja on käytettävissä missä tahansa tiedoston aliohjelmassa tai funktiossa. Globaalit muuttujat ovat myös tuettuja. Globaalia muuttujaa pystyy käyttämään moduulin ulkopuolisessa tiedostossa. [1, s. 27-28.]

Cognos-makrokieli on vahvasti tyypitetty ohjelmointikieli. Tuettuja tietotyyppejä ovat numero (*number*), merkkijono (*string*), lista (*array*), joukko (*record*), objekti (*object*) sekä variantti (*variant*). [1, s. 31.]

Numeroita ovat kokonaisluku (*integer*), pitkä kokonaisluku (*long*), yksinkertaisen tarkkuuden liukuluku (*single*), kaksinkertaisen tarkkuuden liukuluku (*double*) ja valuutta (*currency*). Numerot ovat aina etumerkillisiä. Numerot toimivat myös totuusarvoina. Arvo 0 tulkitaan epätodeksi ja muut numeraaliset arvot tulkitaan todeksi. Vertailuoperaatiot palauttavat 0 epätodeksi ja -1 todeksi. [1, s. 32.]

Merkkijono voi olla määrätynkokoinen tai dynaaminen. Sen pituus voi vaihdella 0:sta 32767 merkkiin. Tyyppimuunnokset määrätynkokoisen ja dynaamisen merkkijo-

non välillä ovat mahdollisia. Jos dynaaminen merkkijono on pienempi kuin määrätynkokoinen merkkijono, merkkijonon loppuosa täytetään välilyönneillä. Jos dynaaminen merkkijono on pidempi kuin määrätynkokoinen merkkijono, silloin merkkijono katkaistaan automaattisesti. Huomionarvoista on se, että merkkijono- ja numerotietotyypin välillä ei tehdä implisiittistä tyyppimuunnosta. Mikäli tämä on tarpeen, tulee käyttää erillisiä funktioita: Val tai Str$. Toinen vaihtoehto on käyttää variantteja, jotka mahdollistavat tyyppimuunnokset minkä tahansa tietotyypin kanssa. [1, s. 33.]

Listat voivat olla määrätynkokoisia tai dynaamisia. Jos listan kokoa muuttaa, niin oletuksena lista tyhjätään. Lisäksi huomionarvoista on se, että listan maksimikoko on 60. [1, s. 33.]

Joukko on tietorakenne, joka koostuu yhdestä tai useammasta elementistä. Kullakin elementillä on oma arvonsa. Elementteihin viitataan piste-notaatiolla (muuttujanNimi.elementinNimi). Joukko voi sisältää elementtejä, jotka ovat itsessään joukkoja. [1, s. 32.] Tietorakenne on verrattavissa C-ohjelmointikielen struct-rakenteeseen.

Objekti viittaa jonkin ohjelman lopputuotteeseen, kuten Visio-dokumenttiin. Ohjelmalla on oma joukkonsa ominaisuuksia ja metodeja, jotka mahdollistavat objektin käsittelyn. Ominaisuus on esimerkiksi Excel-alkion koko tai taustaväri. Metodien avulla ohjelma saadaan tekemään toimenpiteitä objektille, kuten tallentamaan työkirja. Ennen kuin objekteja voidaan käsitellä makroissa, tulee löytyä tapa päästä käsiksi ohjelmaan liittyvään objektiin. Kuinka tämä tehdään, riippuu ohjelmasta, sillä objektien, muuttujien ja metodien nimet ovat ohjelmakohtaisia. [1, s. 44-45.]

Variantit voivat sisältää mitä tahansa tietotyyppiä. Lisäksi tietotyyppi voi vaihtua makron ajonaikana. [1, s. 28.] Taulukosta 3.1. nähdään variantin mahdolliset tietotyypit sekä raja-arvot. Raja-arvot pätevät myös numerotietotyyppeihin.

Taulukko 3.1. *Varianttityypit ja raja-arvot [1, s. 34-35].*

Tyyppi	**Mistä**	**Mihin**
Tyhjä / Null	-	-
Kokonaisluku	-32768	32767
Pitkä kokonaisluku	-2147483648	2147483647
Yksinkertaisen tarkkuuden liukuluku	-3,402823e+38 0,0 1,401298e-45	-1,401298e-45 3,402823466e+38
Kaksinkertaisen tarkkuuden liukuluku	-1,797693134862315e+308 0,0 4,94065645841247e-308	-4,94065645841247e-308 1.797693134862315e+308
Valuutta	-922337203685477,5808	922337203685477,5807
Päivämäärä	1.1.100	31.12.9999
Merkkijono	0	n. 64000 merkkiä
Objekti	-	-

Cognos-makroja voidaan kirjoittaa millä tahansa tekstinkäsittelyohjelmalla. Tätä tarkoitusta varten on kuitenkin kehitetty IBM:n CognosScript Editor. Sen käyttäminen helpottaa makron kääntämis- ja testausprosessia. Esimerkiksi sen avulla makroa voi ajaa rivi kerrallaan sekä tarkastella muuttujien arvoja. Jotta makroja voidaan ajaa, ne pitää ensin kääntää. Tämä onnistuu CognosScript Editorin kautta tai komentoriviltä runmac32-ohjelmalla. Kääntämisen yhteydessä makrotiedostosta (.mac) muodostetaan ajotiedosto (.mcx).

4 PROSESSIT JA MAKROT

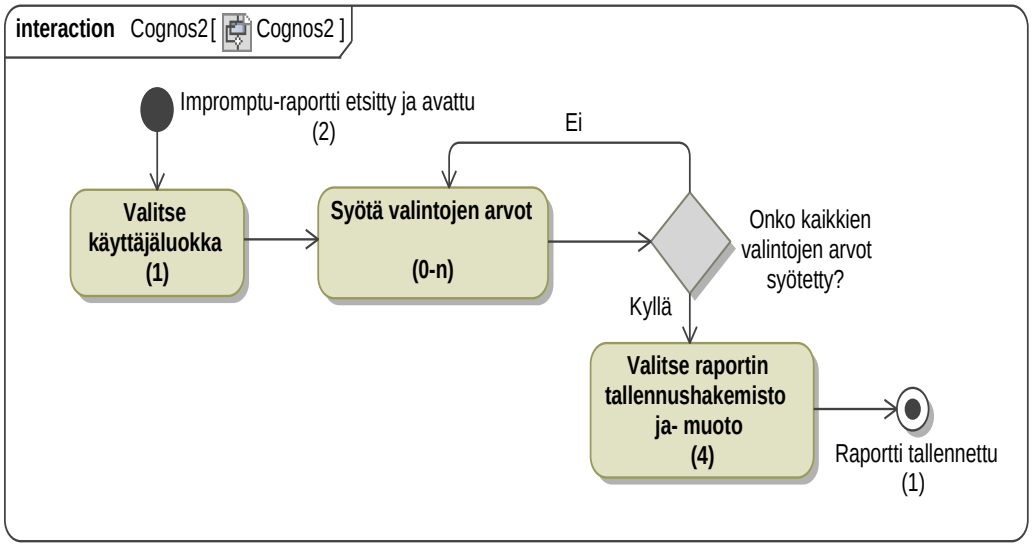
CGI:n yksittäisen tiimin säännöllisiin Cognos Series 7 -tuoteperheen töihin sisältyi vuonna 2011 neljän katalogin päivittäminen listaraporttijulkaisuineen, noin 50 kuution päivittäminen ja julkaisu sekä 80 Excel-raportin toimittaminen asiakkaille joka kuukausi. Näiden ohella töihin lukeutuivat muun muassa tietovarastolataukset, Excel-työkirjojen manuaaliset muokkaukset sekä Cognos 8:n kuutiojulkaisut. Työt oli jaettu kolmen henkilön kesken, ja kiireisin ajankohta oli kuukauden vaihde.

Tietovarastolataukset olivat suurimmalta osin ajastettu ja niiden onnistumista tarvitsi lähinnä valvoa. Kuutioiden muodostukseen mallista oli tehty komentorivikehote. Excel-raporttien toimittamiseen kului eniten aikaa, sillä Impromptu-raportin avaamisen ja mahdollisen parametrien syöttämisen jälkeen piti odottaa, että tietokannasta haetaan kaikki rivitiedot raportille. Riippuen tietomäärästä ja raporttimääritysten monimutkaisuudesta aikaa saattoi kulua muutamasta sekunnista kymmeneen minuuttiin. Tämän jälkeen raportti voitiin viimein tallentaa Excel-muotoon haluttuun sijaintiin. Työmäärästä kertoo se, että 34 Excel-raporttikokonaisuuden toimittamiseen oli varattu kolme työpäivää yhdeltä henkilöltä. Lisäksi kahdessa tapauksessa Excel-työkirjan sisältöä piti muokata manuaalisesti. Esimerkiksi oli tarve nimetä otsikot uudelleen, vaihtaa kenttien järjestystä ja toistaa rivitietoja tietyn logiikan mukaisesti ennen kuin Excel-tiedostoa voisi käyttää tietokantalatauksen tietolähteenä.

Kuukausittaisiin töihin toivottiin helpotuksia, jotta manuaaliset työvaiheet vähenisivät ja työnteko tehostuisi. Lisäksi kuukausittaisten töiden ulkopuolelle lukeutui tehtäviä, jotka olisi hyvä automatisoida. Yksi näistä oli Upfront-verkkoportaalin käyttäjien lisenssitietojen raportointi asiakaskohtaisesti. Cognos-makroilla toteutettuja ratkaisuita käsitellään alla olevissa kohdissa.

4.1 Impromptu-muunnokset

Prosessi Impromptu-raportin manuaalisesta muuntamisesta toiseen tiedostomuotoon on seuraava: Ensiksi avataan haluttu Impromptu-raportti. Siihen liittyy jo valmiiksi katalogi, mutta tulee valita käyttäjäluokka, jona raportti avataan. Tämän jälkeen syötetään mahdolliset valintojen arvot raportoitavan ajanhetken mukaisesti. Raporttiin voi liittyä 0, 1 tai useampi valinta. Sitten odotetaan, että Impromptu-raportti hakee kaikki tiedot tietokannasta. Kun raportin tiedot on haettu, ne näytetään Impromptussa. Lopuksi valitaan kansio, jonne raportti tallennetaan sekä annetaan sille nimi ja määritetään tallennusmuoto. Kuva 4.1. havainnollistaa prosessia. Suluissa numero kuvaa, kuinka monta hiiren klikkausta kukin vaihe edellyttää. Näppäinpainalluksien määrää ei huomioida.



Kuva 4.1. *Työvaiheet Impromptu-raportin muuttamisesta toiseen tiedostomuotoon.*

Vastaavanlainen prosessi on toteutettavissa Cognos-makrolla. Koska käyttötapaus on CGI:llä yleinen, käsitellään muunnokset kolmella eri lähestymistavalla alakohdissa 4.1.1, 4.1.2 sekä 4.1.3.

4.1.1 Xls-muunnos ja päivämäärien päättely

Makron suoritus alkaa Main-nimisestä pääohjelmasta ja päättyy End Sub -komentoon. Pääohjelman ulkopuolelle on mahdollista määrittää aliohjelmia ja funktioita sekä muuttujia. [1, s. 17.]

```
Sub Main()
'Kommentti: Pääohjelman sisältämä koodi tulee tähän.
End Sub
```

Määritetään aluksi makron käyttämät muuttujat avainsanalla Dim. Dim-avainsanaa seuraa muuttujan nimi. Tämän jälkeen As-avainsanalla voidaan määrittää muuttujan tyyppi. Jos tyyppiä ei määritellä, tyypiksi tulee variantti [1, s. 113]:

```
Dim objImpApp as Object
Dim objImpRep as Object
Dim strExcelMovePath as String
Dim strExcelFileName as String
Dim excelDate as String
Dim paydate as String
Dim today as Integer
Dim reportDay as Integer
Dim reportMonth as Integer
Dim reportYear as Integer
```

```
Dim lastDayOfMonth as Integer
```

Seuraavaksi alustetaan strExcelMovePath-muuttuja, joka sisältää tiedon, minne kansioon Excel-raportti tallennetaan. Muuttujat alustetaan yhtäsuuruusmerkillä, ja koska kyse on merkkijonotyyppisestä muuttujasta, käytetään lainausmerkkejä:

```
strExcelMovePath = "T:\Demo\Kyselyt\Demot\Muunnokset\"
```

Oletetaan, että Impromptu-raportin avaamisen yhteydessä tarvitsee syöttää maksupäivä. Lisäksi oletetaan kuukaudessa olevan kaksi erillistä maksupäivää. Toinen on kuukauden ensimmäinen päivä ja toinen on kuukauden 16. päivä. Päätellään makron ajopäivästä kumpaa käytetään. Ajopäivä saadaan selville Day(Now), ajokuukausi Month(Now) sekä ajovuosi Year(Now) -funktioilla:

```
today = Day(Now)
reportMonth = Month(Now)
reportYear = Year(Now)
```

Esimerkin tapauksessa, jos raporttia ajetaan kuukauden alussa, raportoidaan edellistä kuukautta. Tällöin raportointipäivä saa arvon 16. Muussa tapauksessa raportoidaan kuluvaa kuukautta:

```
if (today < 15) then
   reportDay = 16
   'koodia
else
   reportDay = 1
end if
```

Edellistä kuukautta raportoidessa vuodenvaihde on huomioitava erikseen. Jos ajokuukausi on tammikuu, niin raportointikuukausi on joulukuu ja vuotta vähennetään yhdellä. Muussa tapauksessa kuukautta vähennetään yhdellä:

```
if (reportMonth = 1) then
   reportMonth = 12
   reportYear = reportYear - 1
else
   reportMonth = reportMonth - 1
end if
```

Päättelyn jälkeen Impromptu-raportille välitettävä maksupäivä muodostetaan funktiolla DateSerial. Käyttämällä samalla Format-funktiota saadaan päivämäärä halutunlai-

seen muotoon. Esimerkiksi, jos valinnan tyypiksi on määritetty päivä, niin maksupäivä tulee antaa muodossa ”yyyy-mm-dd”:

```
paydate = Format(DateSerial(reportYear, reportMonth,
                  reportDay), "yyyy-mm-dd")
```

Excel-tiedoston nimeen liitettävän päivämäärän päättely ohitetaan tässä, mutta logiikan voi tarkistaa liitteestä 1. Avataan seuraavaksi Impromptu luomalla uusi OLE-automaatio-objekti. Alaviiva ensimmäisen rivin lopussa tarkoittaa sitä, että komento jatkuu seuraavalta riviltä:

```
Set objImpApp = _
 CreateObject("CognosImpromptu.Application")
```

Oletuksena makro suoritetaan taustalla. Jos toimenpiteet haluaa nähdä, onnistuu se komennolla:

```
objImpApp.Visible 1
```

Avataan seuraavaksi raporttiin liittyvä katalogi. Komennon jälkimmäinen parametri määrittää käyttäjäluokan, jona katalogi avataan. Eri käyttäjäluokilla saattaa olla määritetty erilaisia oikeuksia. Esimerkiksi tietty käyttäjäluokka oikeuttaa vain tarkastelemaan tietyn kustannuspaikan tietoja:

```
objImpApp.OpenCatalog _
 "T:\Demo\Katalogit\demo.cat", "Creator"
```

Seuraavaksi avataan Impromptu-raportti. OpenReport-metodin ensimmäinen parametri on polku imr-tiedostoon. Lisäksi kyseinen raportti sisältää yhden valinnan, ja sen saama arvo ilmoitetaan pilkun jälkeen:

```
Set objImpRep = _
objImpApp.OpenReport("T:\Demo\Kyselyt\demot\
          demo_palkkatapahtumat.imr", paydate)
```

Haetaan seuraavaksi tietokannasta kaikki rivit. Komento ei ole pakollinen, jos raportilla on asetus, että tieto haetaan automaattisesti [21]:

```
objImpRep.RetrieveAll
```

Raportin nimen alkuosa tulee Impromptu-tiedoston nimestä. Left-funktion avulla otetaan nimen alusta tietty määrä merkkejä. Ensimmäinen parametri on Impromptu-

tiedoston nimi. Toinen parametri kertoo, kuinka monta merkkiä otetaan. $-merkillä voi ilmoittaa, että paluutyyppi on merkkijono. Jos $-merkkiä ei käytä, funktio palauttaa variantin tyypin merkkijono tai tyhjä. [1, s. 229.] Samalla hyödynnetään Len-funktiota, jonka avulla tiedoston nimestä jätetään pois tiedoston pääte eli ”.imr”:

```
strExcelFileName = Left$(objImpRep.Name, Len _
                   (objImpRep.Name) - 4)
```

Seuraavaksi tallennetaan Impromptu-tiedosto Excel-muotoon. Muuttujaan strExcelMovePath on tallennettu tieto kansiosta, johon tiedosto tallennetaan. Tätä seuraavat käskyt muodostavat tiedoston nimen sekä päätteen. Jos samanniminen tiedosto oli jo olemassa, se ylikirjoitetaan. Huomionarvoista on se, että merkkijonoja yhdistetään &-merkin avulla:

```
objImpRep.ExportExcelWithFormat strExcelMovePath &
          strExcelFileName & " " & excelDate & ".xls"
```

Kun muunnos on tehty, suljetaan raportti sekä katalogi. Jos sama makro sisältää useita muunnoksia ja muunnokset käyttävät samaa katalogia, ei katalogia tarvitse sulkea:

```
objImpRep.CloseReport
objImpApp.CloseCatalog
```

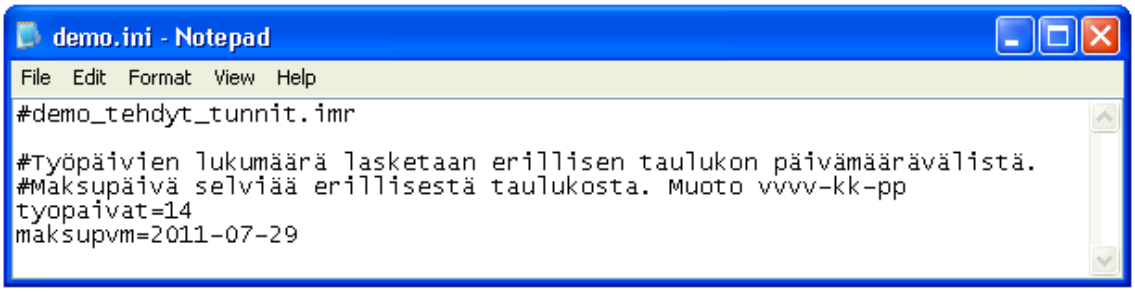
Lopuksi, kun kaikki muunnokset on suoritettu, suljetaan Impromptu. Muuttujat voidaan asettaa myös tyhjiksi:

```
objImpApp.Quit
Set objImpRep = Nothing
Set objImpApp = Nothing
End Sub
```

4.1.2 Iqd-muunnos ja arvojen luku tiedostosta

Aina makrossa ei ole mahdollista päätellä käytettävän parametrin arvoa. Esimerkiksi CGI:llä on käytössä muutama Impromptu-raportti, joissa valinnalle annetaan työpäivien lukumäärä tiettynä ajanjaksona. Ajanjakso tarkistetaan ja työpäivienlukumäärä lasketaan asiakkaan toimittamasta taulukosta. Näissä tapauksissa arvot voidaan tallentaa erilliseen asetustiedostoon ja lukea ne makrossa.

Oletetaan käytössä olevan kuvan 4.2. mukainen asetustiedosto. Risuaitamerkillä # olevat rivit tarkoittavat kommentteja. Tarkoituksena on välittää asetustiedostoon tallennetut arvot Impromptu-raportille.



Kuva 4.2. *Asetustiedosto.*

Aluksi määritetään muuttujat asetustiedostolle, työpäivien lukumäärälle sekä maksupäivälle pääohjelman ulkopuolella. Tällöin muuttujien näkyvyysalue on kaikkialla tiedoston sisällä:

```
Dim settingsFile as String 'asetustiedosto
Dim workdays as String 'työpäivien lukumäärä
Dim paydate as String 'maksupäivä
```

Makro käyttää kahta aliohjelmaa asetustiedoston lukemiseen. Jotta aliohjelmien toteutus voisi sijaita myöhemmin koodissa, aliohjelmat kannattaa esitellä makron alussa:

```
Declare Sub ReadSettings()
Declare Sub GetParameter(currentLine as String,
                         currentParameter as String)
```

Käytettävän asetustiedoston polku luetaan komentoriviltä Command-komennolla:

```
Sub Main()
   'Poistettu muuttujien esittelyt esimerkistä
    settingsFile = Command
```

Seuraavaksi tarkistetaan, löytyykö asetustiedostoa. Jos asetustiedostoa ei ole annettu, settingFile-muuttujan arvo on tyhjä merkkijono. Jälkimmäisellä tarkistuksella tarkistetaan, onko annettu tiedosto olemassa. Ehtojen järjestyksellä on merkitystä, koska Dir-funktio parametrinaan tyhjä merkkijono palauttaa ensimmäisen tiedostonnimen nykyisestä kansiosta [1, s. 116]. Jos tiedostoa ei ole olemassa, annetaan tästä ilmoitus ja hypätään pääohjelman loppuun Exit Sub -komennolla:

```
if settingsFile = "" or Dir(settingsFile) = "" then
    MsgBox "Asetustiedostoa ei annettu/löydy, suoritus
            lopetetaan."
    Exit Sub
end if
```

Seuraavaksi kutsutaan aliohjelmaa asetustiedoston lukemiseksi. Call-komentoa voi käyttää aliohjelmien, funktioiden tai C:llä ohjelmoitujen dll-proseduurien kutsumiseen [1, s. 60]:

```
Call ReadSettings()
```

Tämän jälkeen määritetään ja alustetaan aliohjelman käyttämät muuttujat:

```
Sub ReadSettings()

Dim currentLine as String
Dim currentParameter as String
currentParameter = ""
workdays = ""
paydate = ""
```

Avataan nyt asetustiedosto luettavaksi. Avainsanana toimii Open, jota seuraa käsiteltävä tiedosto. For-avainsanalla määritetään moodi. Käytetään Inputia, jotta luetut arvot saadaan tallennettua muuttujaan. Access-avainsanalla voidaan rajata tapahtuma olemaan luku-, kirjoitustapahtuma tai molemmat. Lock-avainsanalla tapahtumaan voidaan liittää lukko. Esimerkiksi Write määrittää, että muut prosessit eivät saa kirjoittaa tiedostoon tiedoston käsittelyn aikana. Lopuksi As-avainsanalla kerrotaan, mitä kahvaa käytetään. Arvo voi olla väliltä 1-255: [1, s. 271-272.]

```
Open settingsFile For Input Access Read Lock Write As #1
```

Luetaan asetustiedosto loppuun EOF-funktiolla. Funktio saa parametrinaan kahvan numeron [1, s. 157]:

```
Do While Not EOF(1)
'Koodia tiedoston käsittelyyn liittyen.
Loop
```

Line Input-komennolla luetaan kahvana olevasta tiedostosta rivi ja tallennetaan se muuttujaan [1, s. 233-234]:

```
    Line Input #1, currentLine
```

Kuvassa 4.2. nähty asetustiedosto sisältää kommentteja sekä tyhjiä rivejä. Käsitellään vain rivit, joiden pituus on erisuuri kuin 0 sekä rivit, joissa ensimmäinen merkki ei ole risuaita:

```
if(Len(currentLine) <> 0 and
   String(1, currentLine) <> "#") then
   'Koodi muuttujien tallentamiseen.
end if
```

Jos rivi ei ole tyhjä tai kommentoitu, sen oletetaan sisältävän muuttujan arvon. Lisäksi oletetaan, että asetustiedostosta luettavat arvot tulevat tietyssä järjestyksessä. Arvon hakemiseen käytetään GetParameter-aliohjelmaa. Cognos-makroissa suluissa olevat parametrit ovat arvoparametreja ja ei-suluissa viiteparametreja. Arvoparametri säilyttää arvonsa, kun aliohjelma tai funktio palaa kutsujaan. Viiteparametria käytettäessä muuttujan arvo saattaa vaihtua kutsun seurauksena: [1, s. 23.]

```
if (workdays = "") then
   Call GetParameter((currentLine), currentParameter)
   workdays = currentParameter
elseif (paydate = "") then
   Call GetParameter((currentLine), currentParameter)
   paydate = currentParameter
end if
```

GetParameter-aliohjelman toteutuksessa hyödynnetään GetField-funktiota. GetField palauttaa annetun merkkijonon x.:nen alimerkkijonon tiettyjen erotinmerkien mukaan. Jos parametrina annettu numero on suurempi kuin mahdollisten alimerkkijonojen määrä, palautetaan tyhjä merkkijono. [1, s. 193.] Esimerkiksi alla haetaan ensimmäisen yhtäsuurusmerkin jälkeen tulevaa alimerkkijonoa joko toiseen yhtäsuuruusmerkkiin tai rivin loppuun asti:

```
Sub GetParameter(currentLine as String,
                 currentParameter as String)
   currentParameter=GetField(currentLine,2,"=")
End Sub
```

Kun silmukka on suoritettu loppuun, suljetaan tiedosto. Aliohjelman päättymisen jälkeen palataan takaisin pääaliohjelmaan:

```
   Close #1
End Sub
```

Pääohjelmassa avataan Impromptu ja katalogi kuten alakohdassa 4.1.1 on kerrottu. Raportin avaaminen eroaa kuitenkin siten, että valinnalle on välitettävä kaksi parametria. Tällöin valintojen saamat parametrit erotellaan putkimerkeillä valintojen mukaisessa järjestyksessä:

```
   Set objImpRep = _
objImpApp.OpenReport("T:\Demo\Kyselyt\Demot\demo_tehdyt_
                     tunnit.imr", workdays+"|"+paydate)
```

Raportin nimi muodostetaan vastaavanlaisella logiikalla kuin edellä, mutta Name-jäsenen sijaan käytetään FullName-jäsentä. FullName palauttaa Impromptu-tiedoston koko polun:

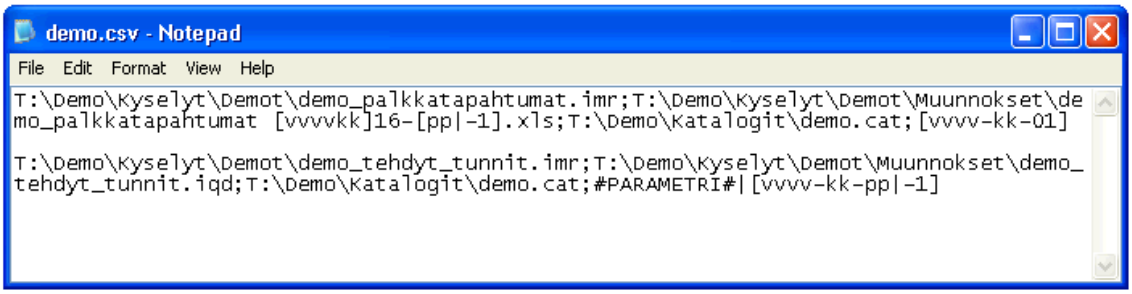
```
   strIQDFileName = Left$(objImpRep.FullName, Len _
                    (objImpRep.FullName) - 4)
```

Tiedoston muuntamiseen iqd-muotoon löytyy oma komentonsa. Komennon jälkeen suoritetaan vastaavat lopettamistoimenpiteet kuin edellä:

```
   objImpRep.ExportTransformer strIQDFileName & ".iqd"
    'Katalogin sulkeminen, raportin sulkeminen…
End Sub
```

4.1.3 Impromptu muunnoksen yleistäminen

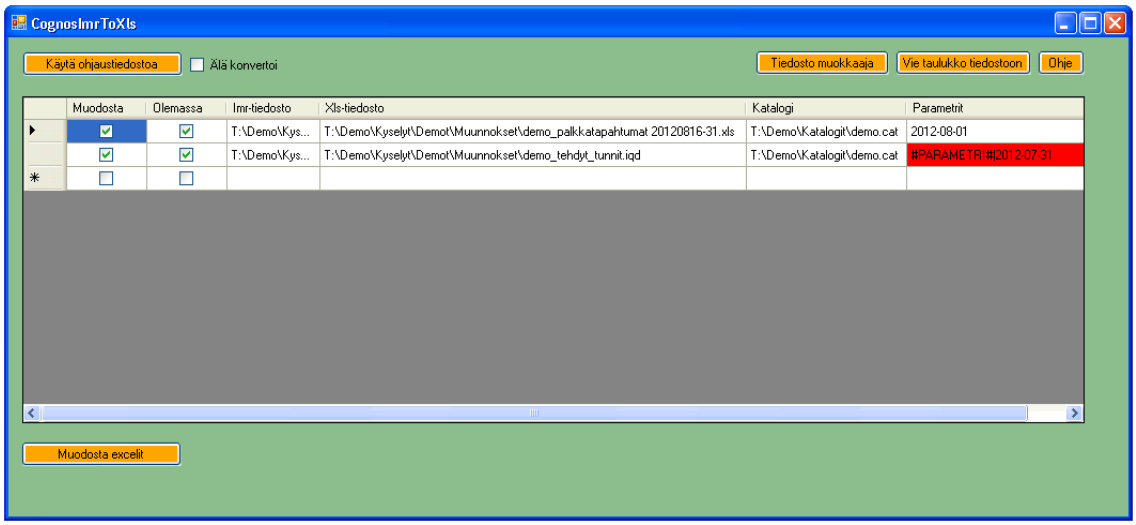
Makrojen ylläpitäminen saattaa olla vaikeaa, mikäli ei ole ohjelmointikokemusta. Ylläpitäminen tulee kyseeseen esimerkiksi silloin, kun tiedoston muodostamispolkua halutaan muuttaa tai muunnokselle välitettävät parametrit muuttuvat. Saattaa myös olla, että uusia muunnoksia tarvitsee lisätä tai vanhoja muunnoksia poistaa koodista. Näiden ongelmakohtien ratkaisemiseksi toteutettiin käyttöliittymä C#:lla, joka vastaa asetustiedoston lukemisesta ja parametrien muuntamisesta oikeaan muotoon. Kuvassa 4.3. nähdään esimerkki asetustiedostosta.



Kuva 4.3. *Käyttöliittymäsovelluksen mukainen asetustiedosto.*

Kuvassa 4.4. nähdään, miltä ohjelma näyttää, kun se on lukenut asetustiedoston. Sarake ”Muodosta” valinta kertoo, tehdäänkö rivin muunnos. Sarakkeen ”Olemassa” ruksi kertoo, että sekä sarakkeessa ”Imr-tiedosto” että ”Katalogi” olevat tiedostot ovat olemassa. ”Xls-tiedosto” sarake määrittää luotavan tiedostonnimen. Sarakkeelle jäi kyseinen nimi, koska ohjelmalla tehdään pääsääntöisesti Excel-muunnoksia. ”Parametrit”-

sarake sisältää muunnoskohtaiset parametrit. Huomiona se, että jos sarakkeen kenttä sisältää sanan ”parametri”, kentän taustaväri on punainen. Tällöin käyttäjä tietää, että arvo on syötettävä manuaalisesti. Ohjelman toteutuslogiikkaan ei mennä tarkemmin, mutta muunnoksien taustalla on yhä Cognos-makrot.



Kuva 4.4. *Käyttöliittymäsovellus käynnissä.*

Makron koko koodi löytyy liitteestä 3, mutta käsitellään piirteet, joita alakohdissa 4.1.1 tai 4.1.2 ei vielä käyty läpi. Pääohjelmasta löytyy virheenkäsittelyä. On Error -komentoa käytetään kertomaan, että jos mikä tahansa virhe tapahtuu, hypätään Err_Main kohtaan. Err_Main vastaa virhetulosteen kirjoittamisesta lokitiedostoon. Kun virhe on käsitelty, Resume-komennolla kerrotaan, mistä kohtaa makro jatkaa toimintaansa. Jos virheitä ei käsitellä, mikä tahansa ajonaikainen virhe johtaa ohjelman suorittamisen lopettamiseen [1, s. 269]:

```
Sub Main()
On Error Goto Err_Main
'koodia
Err_Main: 'hyppykohta
sErrorMessage = "#ERROR " & Err & ": " & Error$ &
                " occurred in Main()"
Print #logFilenumber, sErrorMessage
Resume Err_Quit:
```

Print-komentoa käytetään lokitiedoston kirjoittamiseen. Vapaan kahvan saamiseksi hyödynnetään FreeFile-funktiota. Se palauttaa pienimmän vapaan kahvan numeron [1, s. 185]. Funktio soveltuu tilanteisiin, joissa ei ole varmuutta, mikä kahva on vapaana:

```
Dim logFilenumber As Integer
logFilenumber=FreeFile
```

Makro kirjoittaa päiväkohtaista lokia, joka sijaitsee logs-kansiossa. Logs-kansion määrittelyyn on käytetty vakiota. Jos tyyppiä ei ilmoiteta, se päätellään lausekkeesta [1, s. 81]. Päiväkohtaisen lokitiedoston saa muodostettua Now-funktion avulla:

```
CONST cLogPath = ".\logs\"
logFile = cLogPath & "log_" &
          Format(DateSerial(Year(Now), Month(Now),
          Day(Now)), "yyyymmdd") & ".txt"
```

Alakohdassa 4.1.2 käsiteltiin lokitiedoston avaamista. Kun moodiksi valitaan Append, kirjoitustapahtumat menevät tiedoston lopun jatkoksi:

```
Open logFile For Append As filenumber
```

Makrolle välitetään komentoriviltä määrämuotoinen parametri. GetField-funktion avulla parametrista pilkotaan Impromptu-tiedosto, muunnostiedosto, katalogi sekä parametrien arvot:

```
cmdLine = Command
imrFile = GetField(cmdLine,1,";")
conversionFile = GetField(cmdLine,2,";")
catalogFile = GetField(cmdLine,3,";")
parameters = GetField(cmdLine,4,";")
```

Muunnostyyppi selviää conversionFile-muuttujasta, josta otetaan Right-funktiolla kolme viimeistä merkkiä. Tulos muutetaan isoiksi kirjaimiksi UCase-funktion avulla, koska tyyppivertailu tehdään myöhemmin isoihin kirjaimiin pohjautuen:

```
conversionType = UCase(Right(conversionFile, 3))
```

Impromptun ja katalogin avaaminen tapahtuu kuten edellä esitettiin. Raportin avaamistapa riippuu siitä, onko raportille välitetty parametreja vai ei:

```
If parameters = "" Then
   Set objImpRep = objImpApp.OpenReport(imrFile)
Else
   Set objImpRep = objImpApp.OpenReport(imrFile,
                   parameters)
End If
```

Tämän jälkeen raportti voidaan muuttaa haluttuun muotoon. Tästä vastaa Convert-aliohjelma. Se saa parametrinaan muunnostyypin, kohdetiedoston sijainnin sekä muuttujan logFilenumber mahdollisen virhetilanteen varalle:

```
Sub convert(conversionType As String, conversionFile As
            String, logFilenumber As Integer)
'koodia
End Sub
```

Convert-aliohjelma tarkistaa ensin, onko muunnostyyppi tuettu. Jos on, yritetään suorittaa muunnos. Jos muunnostyyppiä ei tueta, tallennetaan lokitiedostoon virheilmoitus. Rakenne on laajennettava, koska If-Else -rakenteen väliin voi helposti lisätä uusia muunnostyyppejä:

```
If (conversionType = "XLS") then
   objImpRep.ExportExcelWithFormat conversionFile
Else
   sErrorMessage = "Unsupported conversion method. File
                    extension must be xls, csv or iqd."
   Print #logFilenumber, sErrorMessage
End If
```

Kaiken logiikan olisi mahdollisesti voinut toteuttaa pelkillä Cognos-makroilla. Lopputulos ei kuitenkaan olisi ollut yhtä käytettävä. Lisäksi aikaa olisi kulunut enemmän. Yksi mahdollinen haittapuoli ratkaisussa on kuitenkin se, että ohjelman käyttäminen edellyttää ympäristöltä vähintään .Net Framework 2.0 tukea.

4.2 Kuution luominen ja muokkaus

CGI:llä on eräs Cognos Series 7 -asiakas, jonka säännölliset työtehtävät poikkeavat merkittävästi muista asiakkaista. Kyseistä asiakasta varten rakennettiin C#:lla käyttöliittymä, joka tukee työvaiheiden suorittamista. Käyttöliittymän painikkeiden avulla ajetaan Cognos- ja vba-makroja sekä komentoriviohjelmia.

Yksi asiakkaaseen liittyvä työvaihe edellyttää Excel-muotoisen kirjanpitosiirtotiedoston muodostamista kuution pohjalta. Ennen kuin kuutio voidaan luoda PowerPlay Transformerilla, pitää Impromptu-raportit tallentaa iqd-muotoon. Kuution luomisen jälkeen kuutiota pitää muokata PowerPlay for Windows -työkalulla ja tallentaa Excel-tiedostoksi. Kun Excel-tiedosto on muodostettu, sitä muokataan lisää, jotta sitä voisi käyttää tietokantalatauksen tietolähteenä. Excel-tiedostosta pitää muun muassa poistaa ylimääräisiä rivejä, toistaa rivitietoja sekä muuttaa sarakejärjestystä ja otsikointia. Excel-tiedoston muokkausta varten toteutettiin vba-makro, jonka toteutus sivuutetaan. Käsitellään seuraavaksi, miten kuution luominen ja sen muokkaus onnistuu makrolla.

Esimerkissä ohitetaan muuttujien määrittely, mutta toteutus löytyy kokonaisuudessaan liitteestä 4. Määritetään aluksi hakemistopolku, josta mallitiedosto avataan:

```
strModelPath = "T:\Demo\Mallit\Demokuutio.pyi"
```

Avataan seuraavaksi PowerPlay Transformer viittaamalla OLE-automaatio-objektiin:

```
Set objTransApp = _
    CreateObject("CognosTransformer.Application")
```

Tämän jälkeen malli voidaan avata OpenModel-metodia käyttäen:

```
Set objModel = objTransApp.OpenModel(strModelPath)
```

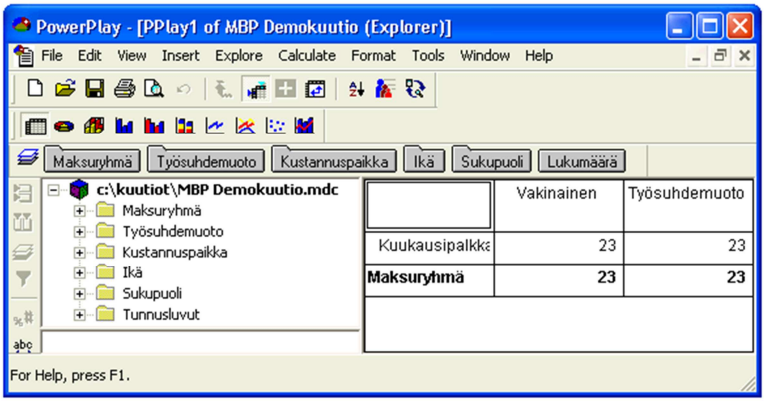
Seuraavaksi valitaan kuutio, joka halutaan luoda. Tämä onnistuu viittaamalla käsiteltävän mallin kuutiojäsenen Item-metodiin joko järjestysnumerolla tai kuution nimellä. Esimerkiksi kuvassa 3.7. nähtiin PowerPlay Transformer -näkymä. Siinä kuutioon voi viitata joko järjestysnumerolla 1 tai nimellä ”MBP Demokuutio”. Kuutio sisältää tiedon muun muassa luotavan kuution hakemistosta ja nimestä sekä listan mittareista, joita kuution käyttäjä voi valita:

```
Set objCube = objModel.Cubes.Item(1)
```

Lopuksi luodaan kuutio CreateMDCFile-metodilla. Metodi soveltuu yhden kuution tai kuutioryhmän kaikkien kuutioiden luomiseen [22, s. 606]:

```
objCube.CreateMDCFile
```

Kun kuutio on luotu, se näyttää kuvan 4.5. mukaiselta. Muokataan seuraavaksi kuutiota makron avulla.



Kuva 4.5. *Kuutio avattuna PowerPlay for Windowsilla.*

Aluksi luodaan kaksi uutta OLE-automaatio-objektia. Ensimmäistä tarvitaan, jotta Cognos PowerPlay for Windows -työkalun voi sulkea. Jälkimmäistä käytetään kuutiomuutosten tekemiseen:

```
Set objPPlayApp = _
     CreateObject("CognosPowerPlay.Application")
Set objPPRep = CreateObject("PowerPlay.Report")
```

Seuraavaksi avataan uusi raporttiobjekti käyttämällä New-metodia. Parametrina annetaan kuutio, joka halutaan liittää raporttiin [23, s. 203]:

```
objPPRep.New "c:\kuutiot\MBP Demokuutio.mdc"
```

Tarkoituksena on lisätä kuutioon sisäkkäinen sarake, joka sisältää kustannuspaikka-dimension kategoriat. Manuaalisesti tämä tapahtuisi raahaamalla kustannuspaikka-dimensio kuution vakinainen ja työsuhdemuoto -sarakkeiden alapuolelle. Jotta kuutiolle voidaan lisätä kategorioita makron avulla, tulee ensin luoda CategoryList-objekti kutsumalla CategoryList-metodia [23, s. 22]:

```
Set objCubeCategories = objPPRep.CategoryList()
```

Tämän jälkeen CategoryList-objektin Add-metodilla valitaan käsiteltävän dimension nimi. Ensimmäinen parametri määrittää tason, josta etsitään lisättävät kategoriat. Tässä käytetään vakiota level_0, jonka arvo on 0. Arvo nolla viittaa määritettyyn dimensioon tai kategoriaan: [23, s. 98.]

```
objCubeCategories.Add level_0, "Kustannuspaikka"
```

Sitten lisätään kustannuspaikka-kategoriat viittaamalla raporttiobjektin sarakkeiden AddLevel-metodiin. Ensimmäisenä parametrina on kokoelma, joka lisätään annettuun dimensioon. Toinen parametri määrittää sisäkkäistason, johon kategoriat lisätään. Kolmas parametri määrittää tason. Jos arvo on 1, kyseessä on ryhmätaso. Arvoa 0 käytetään sisäkkäistason lisäämiseksi. Vakion add_to_all arvo on 1. Neljäs parametri määrittää, lisätäänkö kategoriat isä- vai lapsitasolle: [23, s. 114.]

```
objPPRep.Columns.AddLevel objCubeCategories, _
                level_0, add_to_all, as_child
```

Seuraavaksi lisätään uusi rivi, joka sisältää sukupuoli-dimension kategoriat. Tämä tehdään kuten edellä, mutta viitataan raporttiobjektin rivien AddLevel-metodiin. Lisäksi, koska kategoriat lisätään rivin ensimmäisiksi, tiedot lisätään isätasolle:

```
objCubeCategories.Add level_0, "Sukupuoli"
objPPRep.Rows.AddLevel objCubeCategories, level_0, _
                add_to_all, as_parent
```

Poistetaan seuraavaksi kuution uloin sarake sekä sisäkkäisin rivi. Tämä onnistuu RemoveLevel-metodilla. RemoveLevel-metodin parametrin arvo 0 viittaa sisäkkäisimpään tasoon. Arvo 1 viittaa seuraavaksi sisäkkäisimpään tasoon ja niin edelleen: [23, s. 230.]

```
objPPRep.Columns.RemoveLevel(1)
objPPRep.Rows.RemoveLevel(0)
```

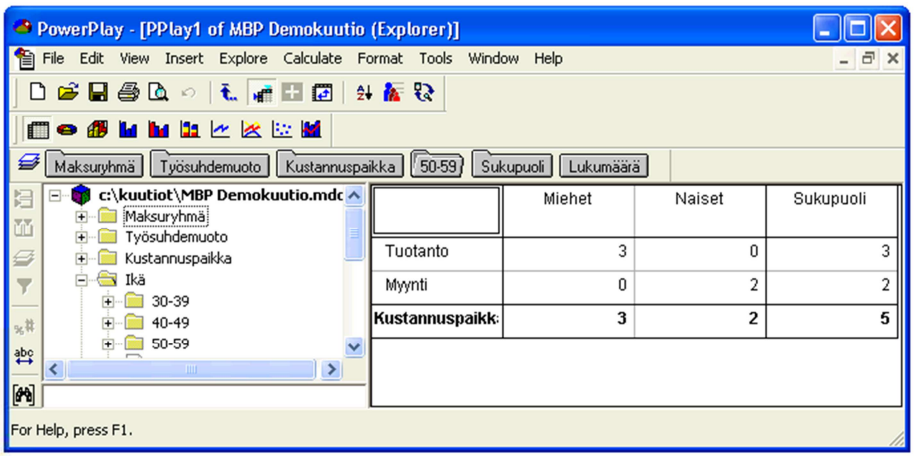
Tämän jälkeen vaihdetaan rivien ja sarakkeiden järjestystä SwapRowsAndColumns-metodilla:

```
objPPRep.SwapRowsAndColumns
```

Sitten suodatetaan tulosjoukkoa siten, että henkilöiden lukumäärään lasketaan ainoastaan henkilöt, joiden ikä on väliltä 50-59. Lisäksi suodatetaan nollarivit raporttiobjektin SuppressZeros-jäsenen avulla. Arvo 3 määrittää, että suodatus koskee ainoastaan rivejä [23, s. 439]. Vaikka CGI:llä ajettavan tietokantalatauksen näkökulmasta nollarivit eivät vaikuta lopputulokseen, niin tietokantalataus on hieman nopeampaa, kun turhat rivit on suodatettu pois:

```
objPPRep.DimensionLine.Item("Ikä").Change("50-59")
objPPRep.SuppressZeros(3)
```

Suodatuksen jälkeen kuutio on kuvan 4.6. näköinen.



Kuva 4.6. *Muokattu ja suodatettu kuutio.*

Tallennetaan kuutio lopuksi Excel-tiedostoksi. SaveAs-metodille annetaan ensimmäisenä parametrina tiedoston nimi. Toinen parametri määrittää tallennusformaatin.

Arvo 4 tarkoittaa Excel-tiedostoa. Jos samanniminen Excel-tiedosto on olemassa, se ylikirjoitetaan oletuksena. [23, s. 243.]

```
objPPRep.SaveAs "c:\kuutiot\demo.xls", 4
```

4.3 Access Manager ja lisenssitietojen raportointi

Asiakkaat pyytävät silloin tällöin raportteja Upfront-verkkoportaalin käyttäjistä. Raportteja pyydetään kahdesta syystä. Toinen on se, että ollaan kiinnostuneita tietyn käyttäjän käyttöoikeuksista eli käyttäjätunnukselle myönnetyistä käyttäjäluokista. Toinen tarve liittyy lisenssitietojen raportointiin.

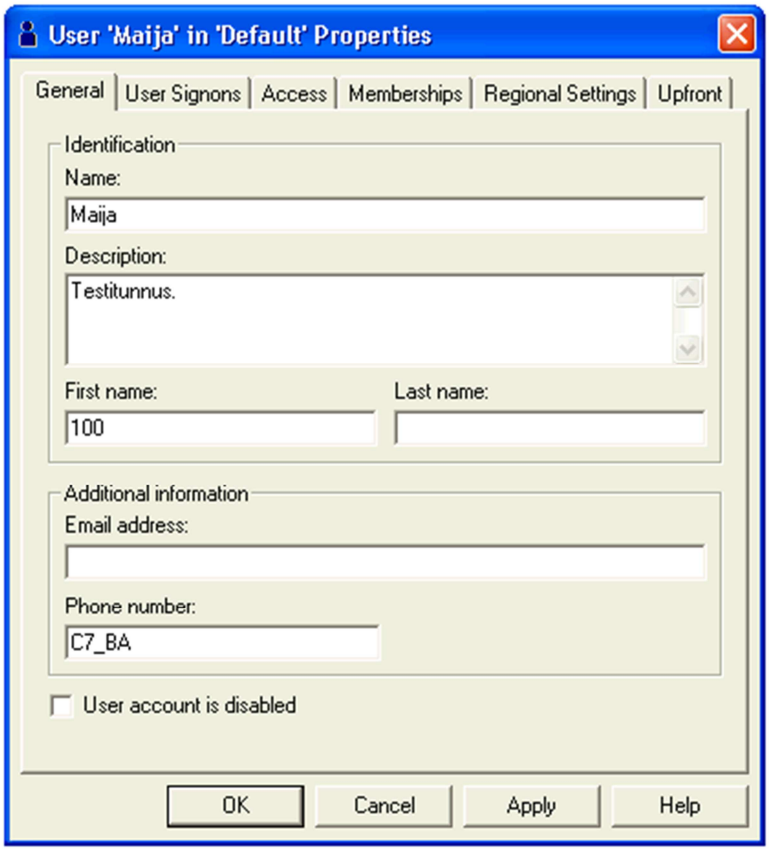
Lisenssitiedot jakautuvat kolmeen ryhmään riippuen myönnetyistä oikeuksista. Joko käyttäjällä on vakioraporttilisenssi, kuutiolisenssi tai molemmat. Vakioraporttilisenssi oikeuttaa käyttäjän tarkastella Upfront-verkkoportaalin vakioraportteja. Kuutiolisenssi lisää oikeuden kuutioihin. Asiakas ei välttämättä ole kiinnostunut, kellä käyttäjällä on mikäkin lisenssi, vaan tarve on tietää lisenssien yhteislukumäärä. Lisäksi yksi asiakas haluaa lisenssien yhteislukumäärän ilmoitettavan maksuryhmäkohtaisesti.

Aiemmin käyttäjätunnuksista pidettiin listaa Excelissä. Excel sisälsi tiedon henkilön nimestä, käyttäjätunnuksen nimestä, käyttäjätunnuksen aktiivisuudesta, käyttäjätunnukseen liittyvistä käyttäjäluokista, maksuryhmästä sekä lisenssitiedon. Näitä tietoja piti ylläpitää Excelissä, kun käyttäjätunnuksen loi, poisti tai muokkasi olemassa olevaa.

Ratkaisuna Excelin manuaaliseen ylläpitoon toteutettiin makro, joka lukee vastaavat tiedot Access Managerista. Toteutus on noin 800 koodiriviä pitkä, joten asiaa käsitellään vain ideatasolla. Toteutuksen pohjana oli kuitenkin jo makro, joka osasi lukea tietoja Access Managerista. Makroa piti kuitenkin kehittää siten, että se pystyisi selvittämään ja laskemaan käyttäjien lisenssitiedot.

Access Managerista löytyy kuvan 4.7. mukainen näkymä. Välilehti General sisältää käyttäjätietoja. Makro suunniteltiin siten, että puhelinnumero-kenttä sisältää lisenssitiedon. Kentän arvona on ”C7_BA”, jos käyttäjälle on oikeus sekä kuutioihin että vakioraportteihin tai ”C7_PP”, mikäli vain kuutioihin. Kentän arvo”C7_IWR” vastaa tilannetta, että käyttäjällä on oikeus vain vakioraportteihin. Lisäksi jos lisenssejä tarvitsee laskea maksuryhmäkohtaisesti, tällöin etunimi-kenttään lisätään maksuryhmän numero. Ylin kenttä eli nimi yksilöi henkilön.

Jotta tietoja voidaan lukea Access Managerista, tarvitaan tätä varten ylläpitäjän käyttäjätunnus ja salasana. Pohjana käytetyssä toteutuksessa tiedot olivat kovakoodattu makroon. Tietoturvallisuuden vuoksi toteutusta muutettiin siten, että tunnus ja salasana kirjoitetaan syöttölaatikkoon. Syöttölaatikon avulla pystyy myös valitsemaan minkä asiakasyrityksen käyttäjätiedot haetaan. Jos mitään ei syötetä, haetaan oletuksena kaikki. Lisäksi makron käynnistämisen yhteydessä voidaan valita, halutaanko käyttäjäluokat hakea. Pelkkien käyttäjäluokkien listaamiseen menee noin 15 minuuttia, eikä niitä voi hakea asiakasyrityskohtaisesti.



Kuva 4.7. *Käyttäjätiedot.*

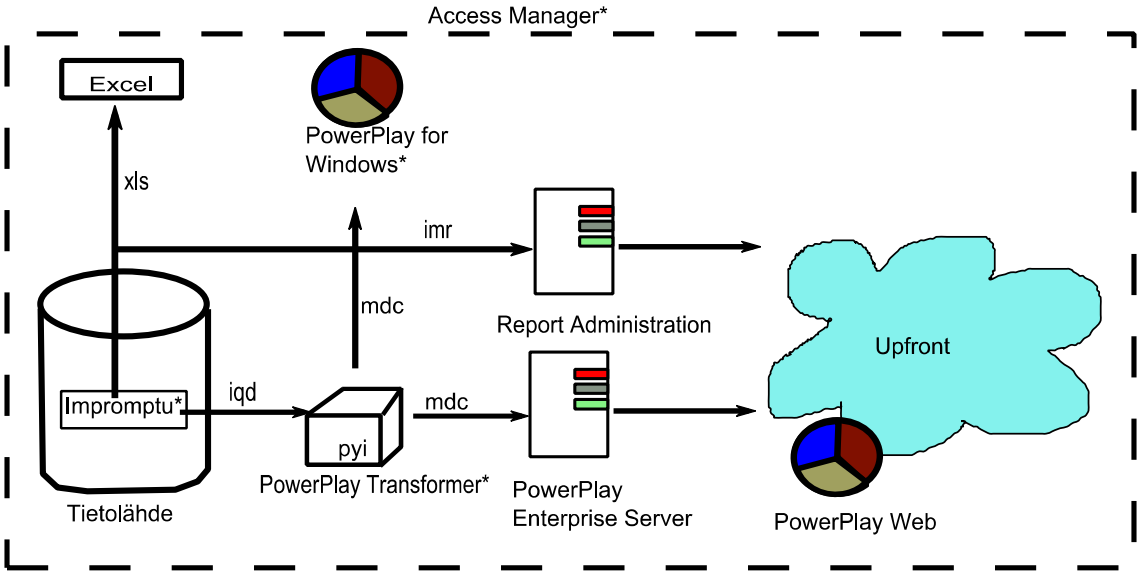
Makron suorittaminen luo kolme csv-tiedostoa. Ensimmäinen tiedosto sisältää Access Manageriin tallennetun henkilön nimen sekä kyseisen käyttäjän käyttäjäluokat. Toinen csv-tiedosto sisältää asiakaskohtaiset tiedot. Sarakkeet ovat: asiakkaan nimi, henkilön nimi, käyttäjätunnus, maksuryhmä, lisenssi, tunnuksen aktiivisuus sekä Access Managerin mukainen polku, josta tiedot on luettu. Maksuryhmä luetaan etunimi-kentästä ja sille etsitään selite erillisestä tekstitiedostosta. Lisenssi-sarakkeen perusteella voidaan tarkistaa, että jokaiselle käyttäjälle on kirjattu lisenssitieto Access Managerin puhelinnumero-kenttään. Tiedosto sisältää myös lisenssitiedot yhteen laskettuina, ja jos lisenssitieto on kirjoitettu väärin, lisenssitieto lasketaan muu-lisenssi alle. Tämän jälkeen käyttäjä voi Excelin lajittelutoimintojen avulla tarkistaa, kenellä henkilöllä on virheellinen lisenssitieto. Kolmas csv-tiedosto sisältää koosteen lisenssitiedoista aktiivisten käyttäjätunnusten osalta. Aktiiviset käyttäjätunnukset ovat niitä, joiden ”User account is disabled”-valinnan kohdalla ei ole ruksia. Tämä tiedosto on sellainen, jonka voi sellaisenaan toimittaa asiakkaalle.

Jos asiakas tarvitsee tiedon käyttäjien käyttäjäluokista, onnistuu tämä yhdistelemällä ensimmäisen ja toisen csv-tiedoston tiedot. Molemmissa tiedostoissa yhdistävänä tekijänä toimii henkilön nimi. Koska käyttäjäluokka csv-tiedosto sisältää henkilöitä kaikista asiakasyrityksistä, ei tietojen yhdistäminen manuaalisesti ole helppoa. Tämän vuoksi tarkoitusta varten toteutettiin C#-ohjelma, joka osaa yhdistää tiedot.

Lopulliseen toteutukseen jäi kuitenkin yksi ongelmakohta, kun lisenssitietoja ilmoittaa maksuryhmittäin. Koska listan maksimipituus on 60 merkkiä, eroavia maksuryhmiä ei saa olla tämän enempää. Käytännössä tämä ei ole ongelma, mutta tästä johtuen etunimi-kentän arvo tulee olla tyhjä, ellei se sisällä maksuryhmätietoa. Toinen opittu asia toteutuksessa oli se, että mikäli Access Managerin kenttää halutaan lukea, tulee muistaa lainausmerkit. Tätä varten makroon lisättiin Quote-funktio. Jos lainausmerkit unohtuvat, kentän arvoa ollaan muokkaamassa, mikä voi olla tuotannossa melko tuhoisaa.

5 ARVIOINTI

Diplomityön lähtökohtana oli löytää keino, miten säännöllisiä Cognos Series 7 -ympäristöön liittyviä töitä voisi automatisoida. Tähän ongelmaan löytyi ratkaisuna Cognos-makrot. Cognos-makroja hyödynnettiin kuvassa 5.1 tähdellä merkittyjen työkalujen yhteydessä eli Impromptussa, PowerPlay Transformerissa, PowerPlay for Windowsissa sekä Access Managerissa.



Kuva 5.1. *Cognos Series 7 –ympäristö ja makrojen käyttö.*

Seuraavissa kohdissa tarkastellaan automatisoinnin onnistumista työkalukohtaisesti.

5.1 Impromptu-muunnoksien onnistuminen

Aluksi muunnosmakroja luotiin kullekin raportointikokonaisuudelle. Yksittäinen makrotiedosto sisälsi kaikki muunnokset, joiden raportit tallennettiin samaan kansioon. Makrojen jaottelun kannalta tämä oli hyvä ratkaisu. Tueksi kirjoitettiin kattava dokumentti, josta voisi tarkistaa makroon liittyvät tiedot: käsiteltävä Impromptu-raportti, sille välitettävät parametrit ja tallennuskansio. Makrojen järkevä nimeäminen tuki myös muistettavuutta. Ratkaisun huonona puolena oli se, että makrojen ylläpito saattaisi osoittautua haasteelliseksi. Jos raportin tallennuskansio tai raportille välitettävät parametrit muuttuisivat, täytyisi muokata makroa. Saattaa myös olla, että makrosta pitäisi poistaa jokin yksittäinen muunnos tai lisätä väliin uusi. Uuden muunnoksen lisääminen kasvattaa makron sisältämää kopioi-liimaa -koodia, mikä heikentää ylläpidettävyyttä. Vaikka näihin muunnosmakroihin oltiin tyytyväisiä eikä makrojen muokkaustarve ole kovin yleinen, ilmennyt ongelma päätettiin ratkaista. Ylläpidettävyyttä varten Impromptu-

muunnosmakrot yleistettiin käyttöliittymän avulla. Sen myötä on alhaisempi kynnys lähteä muokkaamaan muunnoksia, mikäli makrokieli on vieras. Lisäksi on pienempi todennäköisyys rikkoa makro. Käyttöliittymä tukee myös ajantasaista dokumentaatiota, sillä siitä näkee suoraan, mitä Impromptu-raportteja muunnoksessa käsitellään. Jos muunnokset olisi suunniteltu näin tehtäväksi heti, aikaa olisi säästetty toteutuksessa.

Ensimmäiset makrot toteutettiin kokeiluluonteisesti. Niissä ei välttämättä noudatettu hyviä ohjelmointikäytäntöjä. Esimerkiksi virheenkäsittely puuttui kokonaan. Toisaalta muunnosmakrot ovat sen verran yksinkertaisia, ettei toiminnan pitäisi muuttua makron ajosta toiseen. Osa makroista sisältää kuitenkin logiikkaa raportointikuukauden päättelyyn. Tällöin pitää muistaa huomioida erikoistapauksena vuoden vaihtuminen. Toteutuksen yhteydessä tämä tilanne muistettiin testata.

Ensimmäisten makrojen toteutuksessa ei myöskään huomioitu lokin muodostusta. Makron ajamisen jälkeen ei voitu tarkistaa, mitä parametreja makro oli välittänyt Impromptu-raportille. Käyttöliittymän myötä lokitiedoston muodostus lisättiin. Tämän yhtenä etuna on se, että kuukausia myöhemmin pystyy tarkistamaan, millä parametreilla raportti muodostettiin. Jos parametreissa epäillään olleen jotain virheellistä, sen voi tarkistaa lokista.

Seurattavuuden lisäksi CGI hyötyy makroista siten, että niiden avulla säännöllisten työtehtävien suorittaminen on nopeampaa. Laskutettava työ voidaan tehdä pienemmällä työpanoksella, mutta laskutettavuusaste voidaan säilyttää samana. Esimerkiksi automatisoinnin myötä 34 Excel-raportin muodostaminen vaatii työntekijän aikaa noin tunnin. Työntekijän vastuulle jää käynnistää muunnos, tarkistaa Excel-tiedostojen toimivuus sekä lähettää sähköpostia asiakkaalle raporttien valmistumisesta. Raporttien muodostuksen aikana työntekijä voi tehdä muita työtehtäviä. Aiemmin todettiin, että samaiseen työtehtävään oli varattu aikaa kolme päivää. Lisäksi todettiin, että säännöllisiä työtehtäviä varten tarvittiin kolmen työntekijän panosta. Automatisoinnin myötä työmäärä pysyy kohtuullisena, vaikka kuukauden vaihteen töitä tekisi vain kaksi henkilöä.

Automatisointi mahdollistaisi myös työtehtävien suorittamisen ajastetusti. Testaamisen yhteydessä kuitenkin havaittiin, etteivät ajastukset toimi, ellei käyttäjä ole kirjautunut palvelimelle. Tämän vuoksi käyttöliittymään ei toteutettu tukea sen kutsumiseksi komentoriviltä.

5.2 Kuution luonti- ja muokkausmakron onnistuminen

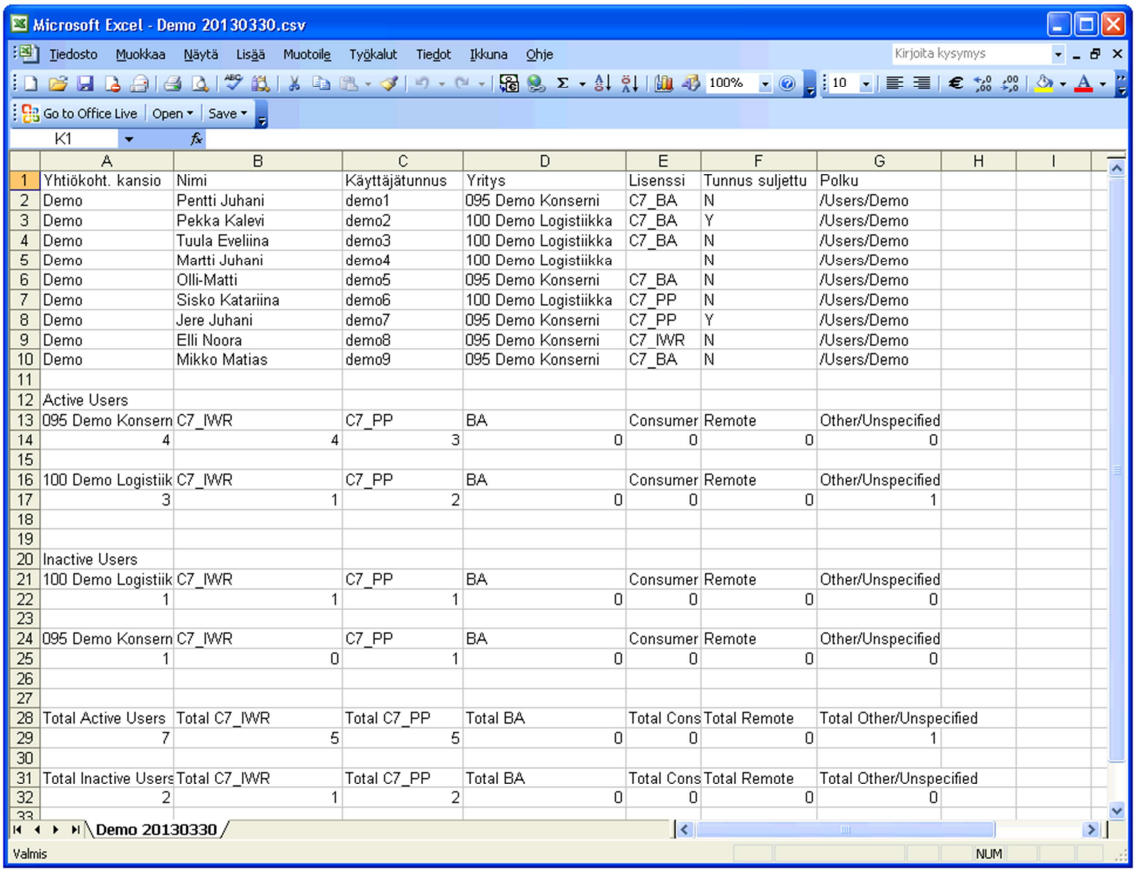
Kuution luonti- ja muokkausmakro tehtiin räätälöidysti yhden asiakkaan säännöllisiä työtehtäviä varten. Arviolta makron toteutukseen meni muutama tunti. Manuaalisesti saman työvaiheen suorittamiseen menee muutama minuutti. Jos Cognos-makroa tarvitaan kerran kuukaudessa, niin ajallisesti siitä saatava hyöty ei ole merkittävä. Samaan työvaiheeseen liittyvästä vba-makrosta saa suuremman hyödyn. Kokonaisuuden kannalta on kuitenkin hyvä, että molemmat työvaiheet voidaan suorittaa käyttöliittymän kautta. Tällöin esimerkiksi uuden työntekijän on helpompi muistaa ja suorittaa kaikki asiakkaaseen liittyvät työtehtävät.

Cognos-makron olisi ehkä voinut kommentoida hieman paremmin. Kyseistä makroa piti ylläpitää vuosi sen toteuttamisen jälkeen, ja olisi ollut eduksi, jos kommenteista olisi selvinnyt mihin kohtaan kuutiota mikäkin dimension lisäys kohdistuu. Lisäksi muokkauksen jälkeen olisi ollut hyvä lisätä makroon kommentiksi versionumero. Jos samalta palvelimelta löytyy samasta makrosta useita versioita, makroa tutkimalla saisi nopeasti selville, mikä on uusin versio ja miksi muutoksia on tehty.

5.3 Lisenssimakron onnistuminen

Makro lisenssitietojen raportointiin toteutettiin olemassa olevan makron pohjalta. Aluksi selvitettiin, miten makro oli toteutettu ajamalla sitä vaihe vaiheelta. Kohtuullisen nopeasti selvisi, mihin kohtiin muutoksia ja lisäyksiä pitäisi tehdä, jotta lisenssitietoja voisi laskea. Koska Access Manager on käytössä myös Cognos 8:ssa, toteutuksessa huomioitiin mahdollisuus laskea myös Cognos 8:sin lisenssitietoja.

CGI hyötyy lisenssimakrosta siten, että sen myötä lisenssitietoja ei tarvitse enää ylläpitää Excelissä. Kun uuden käyttäjätunnuksen luo Access Managerilla, on todennäköisempää, että samalla lisenssitieto muistetaan merkitä. Ja mikäli ei muisteta, tämä havaitaan viimeistään lisenssimakron ajon yhteydessä. Esimerkiksi kuvasta 5.2. havaitaan, että henkilön Martti Juhani lisenssitieto on unohtunut merkitä. Lisenssitiedot pysyvät makron myötä varmemmin ajan tasalla.


Microsoft Excel - Demo 20130330.csv

Tiedosto Muokkaa Näytä Lisää Muotoile Työkalut Tiedot Ikkuna Ohje

Go to Office Live | Open | Save

K1

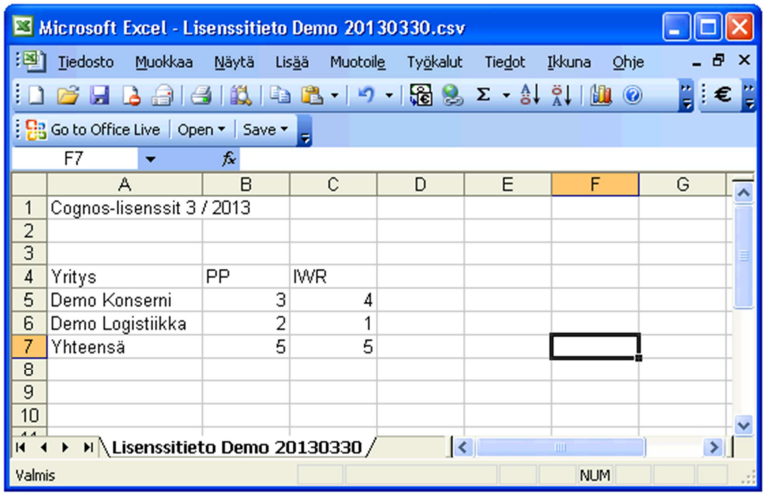
	A	B	C	D	E	F	G	H	I
1	Yhtiökoht. kansio	Nimi	Käyttäjätunnus	Yritys	Lisenssi	Tunnus suljettu	Polku		
2	Demo	Pentti Juhani	demo1	095 Demo Konserni	C7_BA	N	/Users/Demo		
3	Demo	Pekka Kalevi	demo2	100 Demo Logistiikka	C7_BA	Y	/Users/Demo		
4	Demo	Tuula Eveliina	demo3	100 Demo Logistiikka	C7_BA	N	/Users/Demo		
5	Demo	Martti Juhani	demo4	100 Demo Logistiikka		N	/Users/Demo		
6	Demo	Olli-Matti	demo5	095 Demo Konserni	C7_BA	N	/Users/Demo		
7	Demo	Sisko Katariina	demo6	100 Demo Logistiikka	C7_PP	N	/Users/Demo		
8	Demo	Jere Juhani	demo7	095 Demo Konserni	C7_PP	Y	/Users/Demo		
9	Demo	Elli Noora	demo8	095 Demo Konserni	C7_IWR	N	/Users/Demo		
10	Demo	Mikko Matias	demo9	095 Demo Konserni	C7_BA	N	/Users/Demo		
11									
12	Active Users								
13	095 Demo Konsern	C7_IWR	C7_PP	BA	Consumer	Remote	Other/Unspecified		
14	4	4	3	0	0	0	0		
15									
16	100 Demo Logistiik	C7_IWR	C7_PP	BA	Consumer	Remote	Other/Unspecified		
17	3	1	2	0	0	0	1		
18									
19									
20	Inactive Users								
21	100 Demo Logistiik	C7_IWR	C7_PP	BA	Consumer	Remote	Other/Unspecified		
22	1	1	1	0	0	0	0		
23									
24	095 Demo Konsern	C7_IWR	C7_PP	BA	Consumer	Remote	Other/Unspecified		
25	1	0	1	0	0	0	0		
26									
27									
28	Total Active Users	Total C7_IWR	Total C7_PP	Total BA	Total Cons	Total Remote	Total Other/Unspecified		
29	7	5	5	0	0	0	1		
30									
31	Total Inactive Users	Total C7_IWR	Total C7_PP	Total BA	Total Cons	Total Remote	Total Other/Unspecified		
32	2	1	2	0	0	0	0		
33									

Demo 20130330

Valmis NUM

Kuva 5.2. *Lisenssimakron tuottama csv-tiedosto Upfront-portaalin käyttäjistä.*

Lisenssitietojen raportointi on myös nopeampaa, sillä makron ajon yhteydessä voi määrittää, kenenkä asiakkaan lisenssitiedot haetaan. Jos samoja tietoja raportoitaisiin Excelin pohjalta, tulisi asiakkaaseen liittyvät rivit kopioida toiseen Excel-tiedostoon ja muokata toimituskelpoiseksi. Makro muodostaa suoraan toimituskelpoisen csv-tiedoston. Esimerkki csv-tiedoston sisällöstä nähdään kuvasta 5.3.



Kuva 5.3. *Esimerkki asiakkaalle toimitettavasta lisenssitietoraportista.*

Lisenssimakron käytöstä pyrittiin tekemään mahdollisimman intuitiivinen. Siitä löytyy kuitenkin yksi kehityskohde. Cognos Series 7 -ja Cognos 8 -ympäristöt sijaitsevat samalla palvelimella ja kummallakin on oma nimitilansa. Jos haluaa raportoida Cognos Series 7 -asiakkaan lisenssitietoja, pitää nimitila olla tämän mukainen tai tarvittaessa vaihtaa manuaalisesti. Makro voisi itse vaihtaa nimitilaa automaattisesti käyttäjän syötteen perusteella. Toisaalta jos makro itse vaihtaisi nimitilaa, ei se välttämättä olisi paras vaihtoehto käytettävyyden kannalta. Ainakin tällöin makro kannattaisi toteuttaa niin, että se ajon jälkeen palauttaa alkuperäisen nimitilan.

6 JOHTOPÄÄTÖKSET

Ohjelmistoalalla tehdään toisteisia työtehtäviä manuaalisesti, vaikka tietokone suoriutuisi niistä tehokkaammin. Tämä saattaa osittain johtua siitä, että ei ole ajateltu, että työtehtävät voisi tehdä toisin tai ei löydy tekijää, joka voisi toteuttaa automatisoinnin. CGI päätti tehostaa omia työprosessejaan ja toivoi helpotuksia IBM:n Cognos Series 7 -ympäristöön liittyvissä säännöllisissä työtehtävissä. Tehokkuutta tavoiteltiin Cognos-makrojen avulla.

Cognos-makrot soveltuvat Cognos Series 7 -ympäristön automatisointiin. Makrojen hyödyntämisessä on kuitenkin se huonopuoli, että kopioi-liimaa koodimäärä kasvaa helposti. Tämä vaikeuttaa automatisointiratkaisun ylläpidettävyyttä. Sen vuoksi Cognos-makrot kannattaa yhdistää johonkin ohjelmointikieleen. Ohjelmointikielellä toteutetun käyttöliittymän vastuulle voidaan jättää asetustiedoston luku ja parametrien välitys makrolle. Ratkaisu parantaa käytettävyyttä, kun käyttöliittymästä nähdään ja sen kautta voi muokata Impromptu-muunnoksessa käytettäviä parametreja.

CGI:lle toteutettavan automatisoinnin ensisijainen tavoite oli vähentää työntekijöiden kuormitusta kuukauden vaihteen töissä. Tähän tavoitteeseen päästiin. Työtehtävät ovat nopeampi suorittaa, ja osa automatisointiratkaisuista tukee muiden työtehtävien suorittamista rinnakkain. Näin on teoreettinen mahdollisuus tehdä enemmän laskutettavaa työtä samassa ajassa. Lisäksi aiemmin säännöllisiin työtehtäviin tarvittiin kolmen työntekijän työpanos. Nykyään työmäärä pysyy kohtuullisena, vaikka työtehtäviä tekisi kaksi henkilöä.

Aiemmin käyttäjistä ja heidän lisenssitiedoistaan pidettiin kirjaa Excelissä. Kun uuden käyttäjätunnuksen loi tai sen käyttöoikeuksia muokkasi, piti muistaa päivittää tiedot Exceliin. Automatisoinnin myötä tähän ei enää ole tarvetta. Makron avulla voi milloin tahansa generoida ajantasaisen listauksen käyttäjistä, heidän oikeuksistaan sekä lisenssitiedoista.

Saadun palautteen perusteella voidaan sanoa, että Cognos Series 7 -ympäristön automatisointi onnistui hyvin. Kehitettävää kuitenkin jäi, sillä osaan työvaiheista tarvitaan yhä työntekijän panosta. Esimerkiksi käyttäjän tulee käynnistää makrot tai suorittaa työvaiheita käyttöliittymän tukemana. Ajastustuki jäi toteuttamatta, koska ajastusten käynnistyminen edellyttää, että käyttäjä on kirjautunut sisään palvelimelle.

Automatisoinnista saatavien hyötyjen ohella diplomityön eräs keskeinen tavoite oli tutustuttaa lukija makrokieleen. Esimerkkien kautta makrojen perusperiaatteet saatiin käsiteltyä. Toivottavasti niistä on apua, mikäli CGI:llä tarvitsee jatkossa ylläpitää toteutettuja makroja.

Diplomityö onnistui hyvin, ja sille asetetut tavoitteet saavutettiin. Mikäli jatkotutkimus tulee ajankohtaiseksi, voisi seuraavaksi tutustua, minkälaisia hyötyjä makroista saisi muissa IBM:n Cognos-sovelluksissa. Cognos Series 7:n jälkeen on julkaistu Cognos 8 ja 10, jotka molemmat tuovat uusia työkaluja raportointikäyttöön.

LÄHTEET

[1] Versio 7.5. Macros and the IBM CognosScript Language. 2010, IBM Corp. 407 p.

[2] Parasuraman, R. & Riley V. Humans and Automation: Use, Misuse, Disuse, Abuse. Human Factors 39(1997)2, pp. 230-253.

[3] Leeming N. Business Process Management Implementation. Journal of Enterprise Architecture (2005)1, pp. 69-91.

[4] Breton R. & Bossé É. The Cognitive Costs and Benefits of Automation. RTO HFM Symposium on "The Role of Humans in Intelligent and Automated Systems", Warsaw, Poland, October 7-9, 2002. RTO-MP-088. pp. 1-12.

[5] Berghammer D. A Cost Benefit Analysis of an Automated Circulation System for a Small Public Library. POSI 5397 Applied Research Project. Texas, 1996. Texas State University-San Marcos, The Department of Political Science. 91 p. + appx 13 p.

[6] Nash, M & Hession S. Justifying an automated library system - a case study. American Society for Information Science Proceedings of the 43rd ASIS Annual Meeting, Anaheim, California, October 5-10, 1980. Chicago: American Library Association, 1980. pp. 80-82.

[7] Bainbridge L. Ironies of Automation*. Automatica 19(1983)6. pp. 775-779.

[8] Endsley, M. R. Automation and Situation Awareness [PDF]. [viitattu 26.1.2013]. Saatavissa: http://satechnologies.com/Papers/pdf/SA%26Auto-Chp.pdf

[9] Endsley M.R. Technological change and individual adjustment. Proceedings of the Human Factors Society 29th Annual Meeting, Baltimore, Maryland, USA, September 29-October 3, 1985. Humans Factors Society. pp. 598-602.

[10] ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M. & Russell, N. Modern Business Process Automation: YAWL and its Support Environment. Springer-Verlag Berlin Heidelberg 2010, Springer. 694 p.

[11] Guide to BPEL [WWW]. de Vos B. & Zwiers J. 28.6.2005 [viitattu 13.4.2013]. Saatavissa: http://www.radikalfx.com/bpel/usage.html

[12] IBM: BPEL Process [WWW]. [viitattu 13.4.2013]. Saatavissa: http://publib.boulder.ibm.com/infocenter/dmndhelp/v6rxmx/index.jsp?topic=/com.ibm.wbit.help.6012.prodovr.doc/topics/cbpelproc.html

[13] Bradford, L. & Dumas, M. Getting Started with YAWL [PDF]. 2007 [viitattu 13.4.2013]. Saatavissa:
http://yawlfoundation.org/yawldocs/GettingStartedWithYAWL.pdf

[14] IBM to Acquire Cognos to Accelerate Information on Demand Business Initiative [WWW]. 12.11.2007 [viitattu 9.4.2012]. Saatavissa:
http://www-03.ibm.com/press/us/en/pressrelease/22572.wss

[15] Kenttälä, M. Impromptu Series7 raportoinnin perusteet. Cognos Oy. Koulutusmateriaali. 58 s.

[16] Versio 7.5. Administration Guide. 2010, IBM Corp. 1024 p.

[17] Versio 7.4. Discovering Transformer. 2006, Cognos Incorporated. 42 p.

[18] Hartley, C.A. Web-based COGNOS PowerPlay Training Manual [PDF]. St. Mary's College of Maryland. [viitattu 20.3.2013]. Saatavissa:
http://www.smcm.edu/cognos/_assets/Training%20Docs/PowerPlay%20Training/PowerPlay%20Training%20Doc.pdf

[19] Versio 7.5. Upfront Server Administrator Guide. 2010, IBM Corp. 60 p.

[20] Versio 7.4. Access Manager Administration Guide. 2006, Cognos Incorporated. 98 p.

[21] IBM: How to disable the Auto Retrieve option within an Impromptu Report [WWW]. 23.4.1998 [viitattu 7.8.2012]. Saatavissa:
http://www-01.ibm.com/support/docview.wss?uid=swg21352684

[22] Versio 7.5. Transformer Macro Reference Guide. 2010, IBM Corp. 925 p.

[23] Versio 7.5. PowerPlay Macro Reference Guide. 2010, IBM Corp. 511 p.

LIITE 1: XLS-MUUNNOS JA PÄIVÄMÄÄRIEN PÄÄTTELY (LÄHDE-KOODI)

```
'Makro muuttaa imr-tiedoston xls-muotoon.

Sub Main()

   Dim objImpApp as Object
   Dim objImpRep as Object

   Dim strExcelMovePath as String 'hakemisto, johon Excel-
   tiedostot halutaan siirtää muunnoksen jälkeen
   Dim strExcelFileName as String 'Raportin nimi
   Dim excelDate as String 'Tiedostonnimeen lisättävä
   päivämääräväli

   Dim paydate as String

   Dim today as Integer
   Dim reportDay as Integer
   Dim reportMonth as Integer
   Dim reportYear as Integer
   Dim lastDayOfMonth as Integer

   'Lopullinen Excel-tiedosto halutaan siirtää tänne
   strExcelMovePath = "T:\Demo\Kyselyt\Demot\Muunnokset\"

   'Selvitä mikä on nykyinen päivä, kuukausi ja vuosi
   today = Day(Now)
   reportMonth = Month(Now)
   reportYear = Year(Now)

   'Ajetaan kuukauden alussa (ajetaan ennen 15. päivää),
    joten kyse on edellisestä kuukaudesta
   if (today < 15) then
      reportDay = 16 'Ajo esim. heinäkuun alussa
      'Ajetaan tammikuussa
      if (reportMonth = 1) then
        'Kuukausi on 12 ja kyse edellisestä vuodesta
         reportMonth = 12
         reportYear = reportYear - 1
      else
         reportMonth = reportMonth - 1
      end if
```

```
'Raporttia ajetaan 15. päivä
else
   reportDay = 1
end if

'Muodostetaan raportille annettava parametri
paydate = Format(DateSerial(reportYear, reportMonth,
                            reportDay), "yyyy-mm-dd")

if reportDay = 1 then
    'Ajetaan kuukauden puolessa välissä
    excelDate = Format(DateSerial(reportYear,
                reportMonth, reportDay), "yyyymmdd") &
                "-15"
else
   'Raportointikuukauden viimeinen päivä

   'Huomioidaan joulukuu erikseen.
   Dim tempYear as Integer
   Dim tempMonth as Integer

   tempYear = reportYear
   tempMonth = reportMonth + 1

   if tempMonth > 12 then
      tempYear = reportYear + 1
      tempMonth = 1
   end if

   lastDayOfMonth = Format(DateSerial(tempYear,
                    tempMonth, 1) - 1, "dd")

   excelDate = Format(DateSerial(reportYear,
               reportMonth, reportDay), "yyyymmdd") &
               "-" & lastDayOfMonth
end if

Set objImpApp = _
 CreateObject("CognosImpromptu.Application")

 'Näytä ruutu
 objImpApp.Visible 1
```

```
    'Avaa katalogi
    objImpApp.OpenCatalog _
    "T:\Demo\Katalogit\demo.cat", "Creator"

    'Valitse imr-tiedosto

    Set objImpRep = _
     objImpApp.OpenReport("T:\Demo\Kyselyt\demot\
               demo_palkkatapahtumat.imr", paydate)

     objImpRep.RetrieveAll

     'XLS-tiedoston muodostus ja siirto.
     strExcelFileName = Left$(objImpRep.Name, Len _
                       (objImpRep.Name) - 4)

     objImpRep.ExportExcelWithFormat strExcelMovePath &
                strExcelFileName & " " & excelDate & ".xls"

     objImpRep.CloseReport
     objImpApp.CloseCatalog

     'Sulje Impromptu
     objImpApp.Quit
     Set objImpRep = Nothing
     Set objImpApp = Nothing
End Sub
```

LIITE 2: IQD-MUUNNOS JA ARVOJEN LUKU TIEDOSTOSTA (LÄHDEKOODI)

```
'Makro muuttaa imr-tiedoston iqd-muotoon.
Dim settingsFile as String 'asetustiedosto
Dim workdays as String 'työpäivien lukumäärä
Dim paydate as String 'maksupäivä

'Aliohjelmat asetustiedoston lukemiseen
Declare Sub ReadSettings()
Declare Sub GetParameter(currentLine as String,
            currentParameter as String)

Sub Main()

   Dim objImpApp as Object
   Dim objImpRep as Object

   Dim strIQDFileName as String 'iqd:n nimi

   'Settings-tiedoston polku otetaan komentoriviltä
   settingsFile = Command

   if settingsFile = "" or Dir(settingsFile) = "" then
      MsgBox "Asetustiedostoa ei annettu/löydy, suoritus
      lopetetaan."
      Exit Sub
   end if

   'Luetaan siirtohakemisto ja raportin parametrit
   Call ReadSettings()

   Set objImpApp = _
    CreateObject("CognosImpromptu.Application")

    'Näytä ruutu
    objImpApp.Visible 1

    'Avaa katalogi
    objImpApp.OpenCatalog _
    "T:\Demo\Katalogit\demo.cat", "Creator"

    'Valitse imr-tiedosto
    Set objImpRep = _
```

```
    objImpApp.OpenReport("T:\Demo\Kyselyt\Demot\demo_
              tehdyt_tunnit.imr", workdays+"|"+paydate)

   'IQD-tiedoston muodostus ja siirto
    strIQDFileName = Left$(objImpRep.FullName, Len _
                     (objImpRep.FullName) - 4)

    objImpRep.ExportTransformer strIQDFileName & ".iqd"

    objImpRep.CloseReport
    objImpApp.CloseCatalog

   'Sulje Impromptu
    objImpApp.Quit
    Set objImpRep = Nothing
    Set objImpApp = Nothing
End Sub

Sub ReadSettings()

    Dim currentLine as String
    Dim currentParameter as String
    currentParameter = ""
    workdays = ""
    paydate = ""

    'Avataan asetustiedosto lukemista varten
    Open settingsFile For Input Access Read Lock Write As #1

    'Luetaan asetustiedosto loppuun saakka ja otetaan
     parametrit talteen.
    Do While Not EOF(1)
       Line Input #1, currentLine

       'Jos tyhjä tai 1. merkki on kommentti, ohitetaan rivi
       if(Len(currentLine) <> 0 and
          String(1, currentLine) <> "#") then
          if (workdays = "") then
             Call GetParameter((currentLine),
                  currentParameter)
             workdays = currentParameter

       elseif (paydate = "") then
```

```
            Call GetParameter((currentLine),
            currentParameter)
            paydate = currentParameter
         end if
      end if

   Loop

   Close #1

End Sub

Sub GetParameter(currentLine as String,
    currentParameter as String)
    currentParameter=GetField(currentLine,2,"=")
End Sub
```

LIITE 3: IMPROMPTU MUUNNOKSIEN YLEISTÄMINEN (LÄHDE-KOODI)

```
Dim objImpApp as Object
Dim objImpRep as Object

CONST cLogPath = ".\logs\" 'Lokitiedoston polku

Declare Sub convert(conversionType As String,
          conversionFile As String, filenumber As Integer)

Sub Main()

   'Virhekäsittely
   On Error Goto Err_Main

   'Tallennetaan tiedot lokitiedostoon
   Dim filenumber As Integer
   filenumber=FreeFile

   Dim logFile As String
   logFile = cLogPath & "log_" & Format(DateSerial(
             Year(Now), Month(Now), Day(Now)),
             "yyyymmdd") & ".txt"

   Open logFile For Append As filenumber
   Print #filenumber, ""
   Print #filenumber, "-------ohjaustiedosto.mac--------"
   Print #filenumber, Time
   Print #filenumber, ""

   'Otetaan komentoriviltä parametrina asetustiedosto
   Dim cmdLine As String
   cmdLine = Command

   'Itse asetustiedoston muoto oletetaan olevan
   '1) imr-tiedosto;
   '2) muunnos-tiedosto;
   '3) käytettävä katalogi;
   '4) annettavat parametrit

   Dim imrFile As String
   Dim conversionFile As String
   Dim catalogFile As String
```

```
Dim parameters As String
Dim conversionType As String 'päättää sen, mitä oikeasti
                              ollaan muodostamassa

imrFile = GetField(cmdLine,1,";")
conversionFile = GetField(cmdLine,2,";")
catalogFile = GetField(cmdLine,3,";")
parameters = GetField(cmdLine,4,";")

'Selvitetään, mitä muunnosta ollaan tekemässä. Merkit
 isoiksi kirjaimiksi.
conversionType = UCase(Right(conversionFile, 3))

Print #filenumber, "IMR: " & imrFile
Print #filenumber, conversionType & ": " &
      conversionFile
Print #filenumber, "KATALOGI: " & catalogFile
Print #filenumber, "PARAMETRIT: " & parameters

Set objImpApp = _
 CreateObject("CognosImpromptu.Application")

' Näytä ruutu
 objImpApp.Visible 1

 ' Avaa katalogi
 objImpApp.OpenCatalog _
 catalogFile, "Creator"

If parameters = "" Then
   'Raportilla ei ole parametreja
   Set objImpRep = objImpApp.OpenReport(imrFile)
Else
   'Raportilla on parametreja
    Set objImpRep = objImpApp.OpenReport(imrFile,
                    parameters)
End If

objImpRep.RetrieveAll
Call convert(conversionType, conversionFile, filenumber)

objImpRep.CloseReport
objImpApp.CloseCatalog
```

```
Err_Quit:

   'Lopetetaan lokin kirjoitus
   Print #filenumber, "-------------end-------------"
   Close #filenumber

   ' Sulje Impromptu
     objImpApp.Quit
     Set objImpRep = Nothing
     Set objImpApp = Nothing
     Exit Sub

Err_Main:
   Dim sErrorMessage As String
   sErrorMessage = "#ERROR " & Err & ": " & Error$ &
                   "occurred in Main()"
   Print #filenumber, sErrorMessage

   Resume Err_Quit:

End Sub

Sub convert(conversionType As String,
            conversionFile As String, filenumber As Integer)

   Dim sErrorMessage As String

   If (conversionType = "XLS") then
      'XLS-tiedoston muodostus ja siirto.
      objImpRep.ExportExcelWithFormat conversionFile

   ElseIf (conversionType = "CSV") then
      'CSV-tiedoston muodostus ja siirto.
      objImpRep.ExportASCII conversionFile

   ElseIf (conversionType = "IQD") then
      'CSV-tiedoston muodostus ja siirto.
      objImpRep.ExportTransformer conversionFile

   Else
      'Unknown conversion method
      sErrorMessage = "Unsupported conversion method. File
```

```
                        extension must be xls, csv or iqd."
        Print #filenumber, sErrorMessage

    End If

End Sub
```

LIITE 4: KUUTION LUOMINEN JA MUOKKAUS (LÄHDEKOODI)

```
Declare Sub CreatePowerCube()
Declare Sub CreateExcel()

Sub Main()
   Call CreatePowerCube()
   Call CreateExcel()
End Sub

Sub CreatePowerCube()

   Dim objTransApp As Object
   Dim objModel As Object
   Dim objCube As Object
   Dim strModelPath As String

   strModelPath = "T:\Demo\Mallit\Demokuutio.pyi"
   Set objTransApp = _
       CreateObject("CognosTransformer.Application")

   Set objModel = objTransApp.OpenModel(strModelPath)
   Set objCube = objModel.Cubes.Item(1)

   objCube.CreateMDCFile

   objModel.Close

   Set objCube = Nothing
   Set objModel = Nothing
   Set objTransApp = Nothing

End Sub

Sub CreateExcel()

   Dim objCubeCategories As Object
   Const level_0 = 0
   Const add_to_all = 1
   Const as_child = 1
   Const as_parent = 0

   'Muokataan kuutiota, mutta tallennetaan se vain xls-
    muodossa
```

```
    Dim objPPRep as Object
    Dim objPPlayApp as Object

    Set objPPlayApp = _
         CreateObject("CognosPowerPlay.Application")
    Set objPPRep = CreateObject("PowerPlay.Report")

    objPPRep.New "c:\kuutiot\MBP Demokuutio.mdc"

    objPPRep.Visible = 1

    Set objCubeCategories = objPPRep.CategoryList()
    objCubeCategories.Add level_0, "Kustannuspaikka"
    objPPRep.Columns.AddLevel objCubeCategories, _
                    level_0, add_to_all, as_child

    objCubeCategories.Add level_0, "Sukupuoli"
    objPPRep.Rows.AddLevel objCubeCategories, level_0, _
                  add_to_all, as_parent
    Set objCubeCategories = Nothing

    objPPRep.Columns.RemoveLevel(1)
    objPPRep.Rows.RemoveLevel(0)

    objPPRep.SwapRowsAndColumns

    objPPRep.DimensionLine.Item("Ikä").Change("50-59")

    objPPRep.SuppressZeros(3)

    'Filename, format, overwrite (True is default)
    objPPRep.SaveAs "c:\kuutiot\demo.xls", 4

    objPPRep.Close
    objPPlayApp.Quit

    Set objPPRep = Nothing
    Set objPPlayApp = Nothing

End Sub
```